Design • Build • Program • Learn • Compete

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

March 2017

The DOBOT Magician

More than just another desktop robot arm

It's a 3D Printer

It's a Laser Engraver

It's a Pick and Placer

And Much More!

Controlling Servos With The CY8CKIT-059

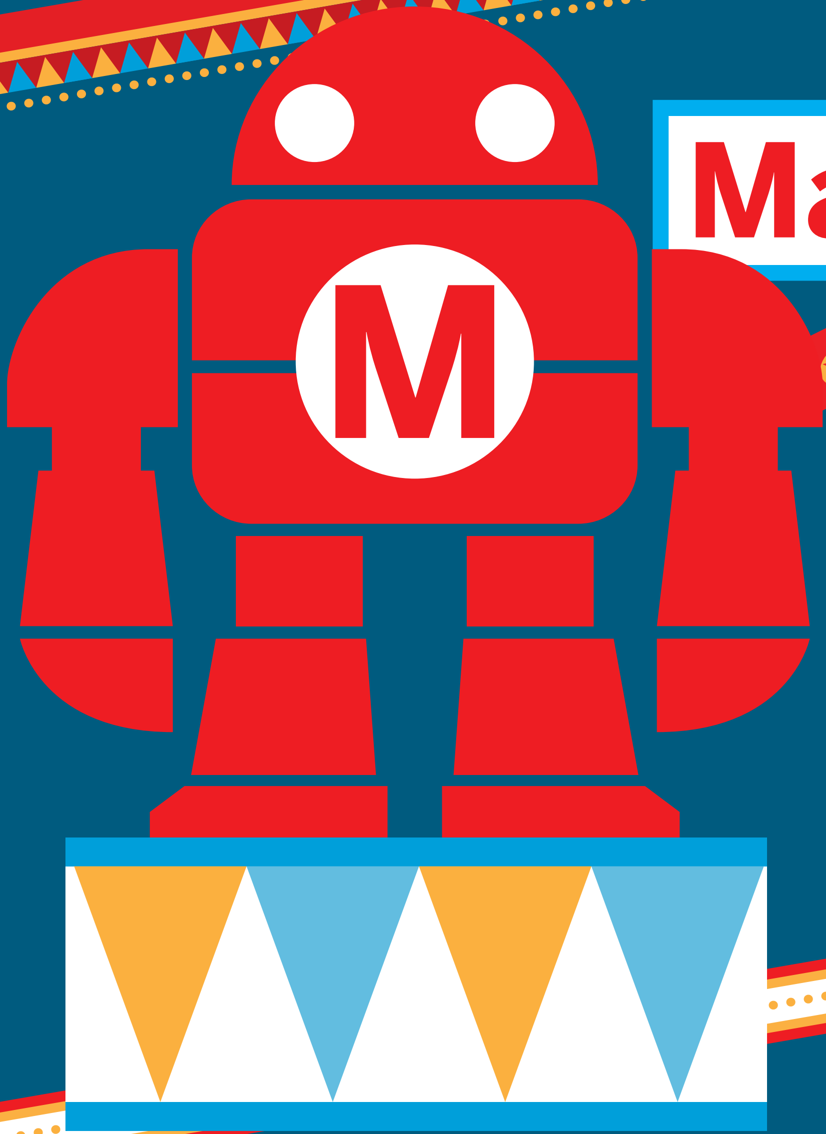
This system on a chip lets you quickly create PWMs as needed

PRESENTED BY intel

Maker Faire®

★ 12TH ★ ANNUAL

BAY AREA



MAY 19–21

SAN MATEO EVENT CENTER

Greatest **SHOW & TELL** *on Earth*

PRESENTED BY intel

GOLDSMITH SPONSORS

Google Microsoft

SILVERSMITH SPONSORS

BOSEbuild Cognizant Digi-Key ELECTRONICS

GET YOUR **TICKETS** TODAY!

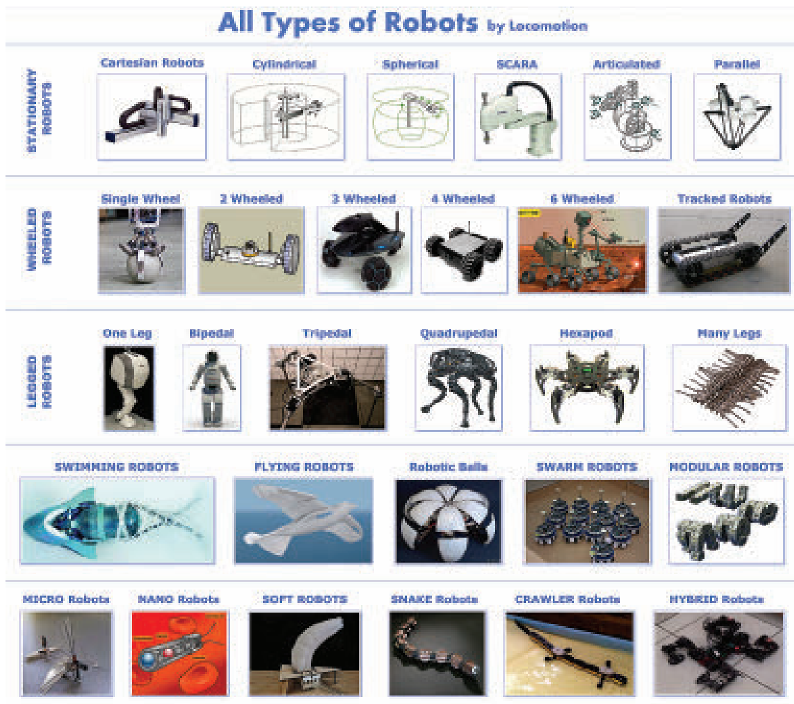
makerfaire.com BROUGHT TO YOU BY Make:

SERVO MAGAZINE

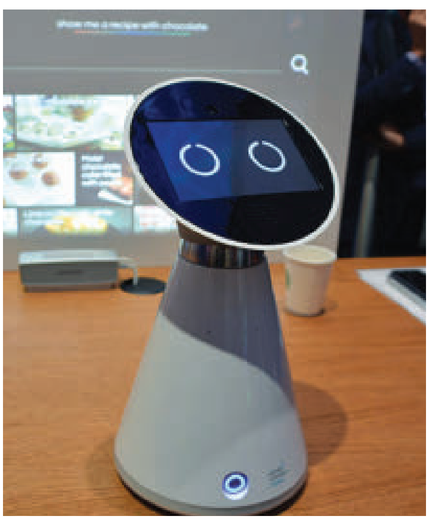
03.2017
VOL. 15 NO. 3

Subscription Information
SERVO Magazine — PO Box 15277
North Hollywood, CA 91615-9218
Call 877-525-2539 or go to www.servomagazine.com
Subscribe • Gift • Renewal • Change of Info

Columns



PAGE 60



The Combat Zone

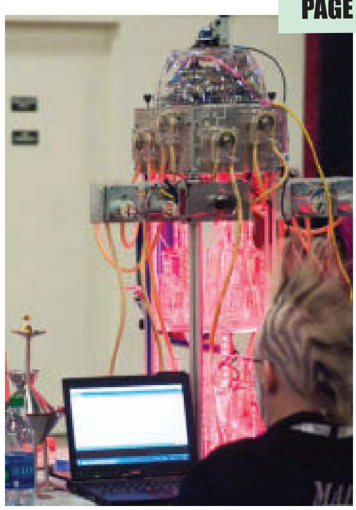
Departments

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

PAGE 52

PAGE 30



Mind / Iron

by Bryan Bergeron, Editor

What's Your Robot Type?

Dog owners are known for selecting a breed with the temperament or personality similar to their own. Based on my experience, I'd go so far as to say that some dog owners even look like their pets. This self-selection behavior makes sense, as most of us like things and people that resemble ourselves. I'm not sure what it reveals about my character, but I'm partial to black labs.

Given the association we have with our pets, it's reasonable to assume that at least some of us prefer robots with particular personalities or even looks. Granted, the sample sizes are small because there isn't as wide of a variety of robot types available in the general consumer market, but some associations seem self-evident.

For example, take the owner of a combat bot. I envision a risk taker, someone a little outrageous in their dress, and someone ready to get down and dirty in the machine shop. Then, there's the owner of an ultra-miniature carpet crawler: a detail-oriented thoughtful type who is more comfortable with a CAD package than with a drill press.

It goes on from here, with personalities mapped to quadcopters and other drones, underwater vehicles, and specialized search and rescue robots.

It might seem like an exercise in silliness, but when it comes to social robots, bonding does matter. It's been shown that the elderly are much more apt to take on a robot designed to monitor their wellbeing if it takes the form of a cuddly seal, for example. Similarly, patients have been

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

shown to more readily accept diagnostic robots when the live video of their physician's head appears on the display that serves as the robot's head.

Fact is, just as with our pets and other people, we have a need to connect with our robots, cars, cell phones, and other devices.

As the number and types of robots increases, I'm sure that our innate preferences will become obvious to the marketing departments of robotics companies, as well as fellow robotics enthusiasts.

Do you already self-identify with a particular type or model of robot? Do you relate best to crafts that move on the ground or glide through the sky? I'm more of a quadcopter type these days, but there's always a next new kind of robot on the horizon. Maybe I'm just always looking for the next new greatest thing.

I suppose a little self-awareness never hurt anyone. **SV**

EVENTS

MARCH

2-5 **Pragyan**
National Institute of Technology, Trichy, India
University level events including Robovigyan.
www.pragyan.org

3-4 **FIRST LEGO League of Central Europe**
Regensburg, Bavaria, Germany
Championship event for FIRST LEGO student teams.
www.first-lego-league.org/en/fll/regions.html

4-5 **METU Robotics Days**
METU Culture & Convention Center, Turkey
Events include Line Follower, Search & Rescue, Trash Hunter, Maze Solver, and an autonomous aerial vehicle contest.
http://odturobotgunleri.org.tr

10-11 **Midwestern Robotics Design Competition**
University of Illinois at Urbana-Champaign, IL
See website for details on this year's event.
http://mrdc.ec.illinois.edu

18 **Manitoba Robot Games**
Tec Voc High School, Winnipeg, Manitoba, Canada
Multiple events including Sumo and Line Following.
www.scmb.mb.ca

23-26 **Apogee**
BITS Pilani KK Birla Goa Campus, Zuarinagar Goa, India
See website for details on this year's events.
www.bits-apogee.org

23-25 **Festival de Robotique**
Montreal, Quebec, Canada
Events include FIRST LEGO, FRC, FLL, and jrFLL.
www.robotiquefirstquebec.org

23-26 **Techkriti RoboGames**
Indian Institute of Technology, Kanpur, Uttar Pradesh, India
University level robot events.
wwww.techkriti.org/#/competitions/robogames

24-26 **Cognizance**
IIT Roorkee, India
University level robot events.
http://cognizance.org.in

26-30 **APEC Micromouse Contest**
Tampa, FL
Micromouse maze running.
www.apec-conf.org

ASK MR. ROBOTO

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

by Eric Ostendorff

Our resident expert on all things robotic is merely an email away.

roboto@servomagazine.com

Q. I have worked my way up through different kinds of robot locomotion and now want to tackle an omni-directional design. I have seen triangular three-wheeled robots that use "omni wheels" to be able to move in any direction.

All my past robotic experiments used a BASIC Stamp 2 as the microcontroller, but I have not been able to find any example source code that would let the Stamp control a three-wheel design.

Do you know if it's possible or do you think I should try a different microcontroller? If so, which one would be best suited to the task?

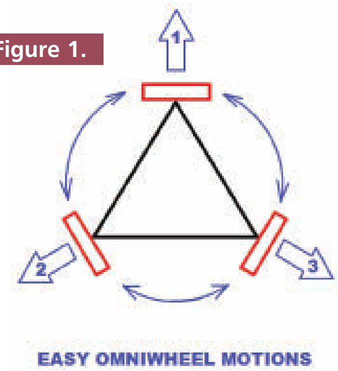
Damion Anderson
Jackson, MS

A. Now 20-some years old, the BASIC Stamp 2 (BS2) remains near and dear to my heart. It's a great educational tool and the heart of tens of thousands of robots and projects. Documentation and code abound for many different types of projects.

It sounds like you have also been a loyal Stamp fan for many moons. Hopefully, you have checked out Parallax's online forum. It's a tremendous resource, chock-full of smart, friendly, helpful, and well-behaved enthusiasts. There's an active robotics sub-forum there, with threads on several omni-wheel bots.

I built a four-wheel mecanum wheel bot powered by a BS2; you can see it in motion at **https://www.youtube.com/watch?v=RPIfIOat6Os**. The DC motors were driven at full speed through a pair of L298N H-bridges, so the programming was very simple for those particular maneuvers. No encoders, no PWM; it was a cinch for the BS2.

Figure 1.



The three omni-wheel bot you describe obviously has different kinematics. A four-wheel omnibot would be easier and more intuitive to program, but if you can live with certain restrictions, the BS2 can definitely drive the three-wheeler you describe. Per **Figure 1**, you can easily spin in place by turning all motors in the same direction.

You can also drive in any of the three primary directions shown by simply driving the opposite two motors at the same speed. Driving one wheel only will give an off-center orbit.

Wheel kinematics become complicated to do much else, and the processor demands increase correspondingly. Mixing wheel directions and speeds is required, and wheel or motor encoders would really help. PWM is not the BS2's strong suit, so I would not recommend ordinary DC motors and H-bridges.

With some clever program crunching and interleaving, the BS2 could drive three continuous rotation (CR) servos. With some persistence and lots of testing, you could experimentally determine the servo values to move in various patterns and write a subroutine for each.

Of course, the BS2 won't have the overhead to do much else while it's driving, yet this would be a serious achievement and certainly appeals to the minimalist in me.

Using a PICAXE would simplify things a bit since it can manage the servo *pulsouts* or PWM for you, but ultimately it has many of the BS2's limitations and would require similar brute force calibration.

If you want to take full advantage of such an omni-wheel platform's excellent maneuverability, you need more number crunching capability for the trig calculations required, and the speed to read wheel encoders. Both the Arduino and the Propeller would be up to the task.

How's your programming in C or Spin? They're certainly a departure from PBASIC in either case. There are examples and code at **http://makezine.com/projects/make-40/kiwi** and **http://forums.parallax.com/discussion/143908**.

The second link is from the very prolific Gareth, who uses a hacked Wowwee Tribot chassis. Quite a stroke of genius in that you can probably find a gently-used ready-to-hack Tribot for less than the price of three new omni wheels. Good luck with your project!

Q. I have been trying to build a robot arm to move player pieces on a board game. I started with an inexpensive robot arm purchased from eBay, and for the life of me I can't get the arm to accurately repeat a series of moves. I found that writing the motion routines to move very slowly helps, but it seems I am unable to get the arm positions to be repeatable. Is there a way to get more accuracy without having to upgrade to a more expensive arm?

Timothy Merchant
Los Angeles, CA

Post comments on this article at
www.servomagazine.com/index.php/magazine/article/March2017_MrRoboto.

A. Any servo based arm should work fairly repeatedly, since the servo joints have potentiometer based position feedback. If you're having consistency issues, I'm guessing that your inexpensive arm does not have feedback. One such arm in my collection is the OWI-535 Robotic Arm Edge shown in **Figure 2**.

It's a beautiful robot and very fun to play with using the wired remote control. There is an optional PC USB interface and program which allows you to "control" the robot by writing scripts or sequences of motions which can be repeated afterward.

Figure 2.

Unfortunately, since there is no feedback, this is an "open-loop" system which can only record timing commands to "repeat" a motion. As simple as it is, it works quite well for general motion playback. However, without positional reference, a known home starting position, and regulated motor speed, attempting to move consistently to precise locations such as game pieces on a board will be nearly impossible.

Even the alkaline batteries introduce errors. As they wear down, motor speeds slow down, and the various arm joints don't travel as far in a fixed timing period. You'll need a servo based arm to get the repeatability you desire.

Q. I will be traveling on business for a few weeks and will be spending a lot of time on airplanes and in motel rooms. As I think the TSA would take a dim view of a carry-on full of wires, printed circuit boards (PCBs) and servo motors, I was wondering if you had any recommendations for fun, useful, or educational podcasts or webcasts I could listen to on my phone.

Larry Ryan
New York, NY

A. There are so many choices given the breadth of today's online offerings! Coursera, Udacity, and **Lynda.com** have both free and paid offerings.

A lesser known free site is **www.learnerstv.com** with lectures on anything imaginable, and you can download the videos there. How fun or useful is all very subjective, of course. Many of those are fairly dry university lectures; definitely educational, but not necessarily entertaining.

Take at least a quick look at them before you end up on a 20 hour flight with nothing but Professor I.M. Boring droning on and draining the fun from your favorite subject.

I used to ride the bus to work, and I always dreamed of building an electronics lab in a briefcase. A small tablet PC and programming cable, breadboard and all the wires, PCBs, and servos you mentioned. I bet there's a market for something like that. Are you sure it's not worth the TSA hassle? Wouldn't it make their day, finding a big red LED countdown clock inside your briefcase? You could certainly use anything you want in your hotel room.

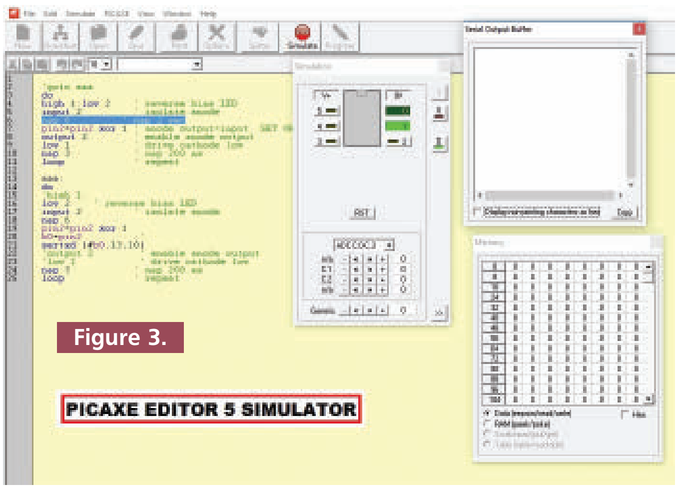


Figure 3.

Maybe on the flight, you could just write code on your carry-on laptop, but carry your wires, servos, and hardware in your checked bags. Depending on which IDE (integrated development environment) you use, you can at least check the syntax of your code.

One of my favorite aspects of the PICAXE programming editor is that it has a built-in simulator which lets you "run" your program without any hardware or programming cable (**Figure 3**). You could certainly do that on a flight without too much hassle. I posted a simulator video at **https://www.youtube.com/watch?v=TftZYLkJcyU**.

The silly program is nothing but *GOTO* commands. It's my tongue-in-cheek flashing light tribute to Edgar Dijkstra, who said that *GOTO*s were "harmful" for programmers and encouraged spaghetti code. Don't know about you, but I've always liked spaghetti!

Speaking of silly program

challenges which might help you while away the lonely hours travelling, this BASIC Stamp 2 program flashes and beeps in the Haiku pattern of 5-7-5.

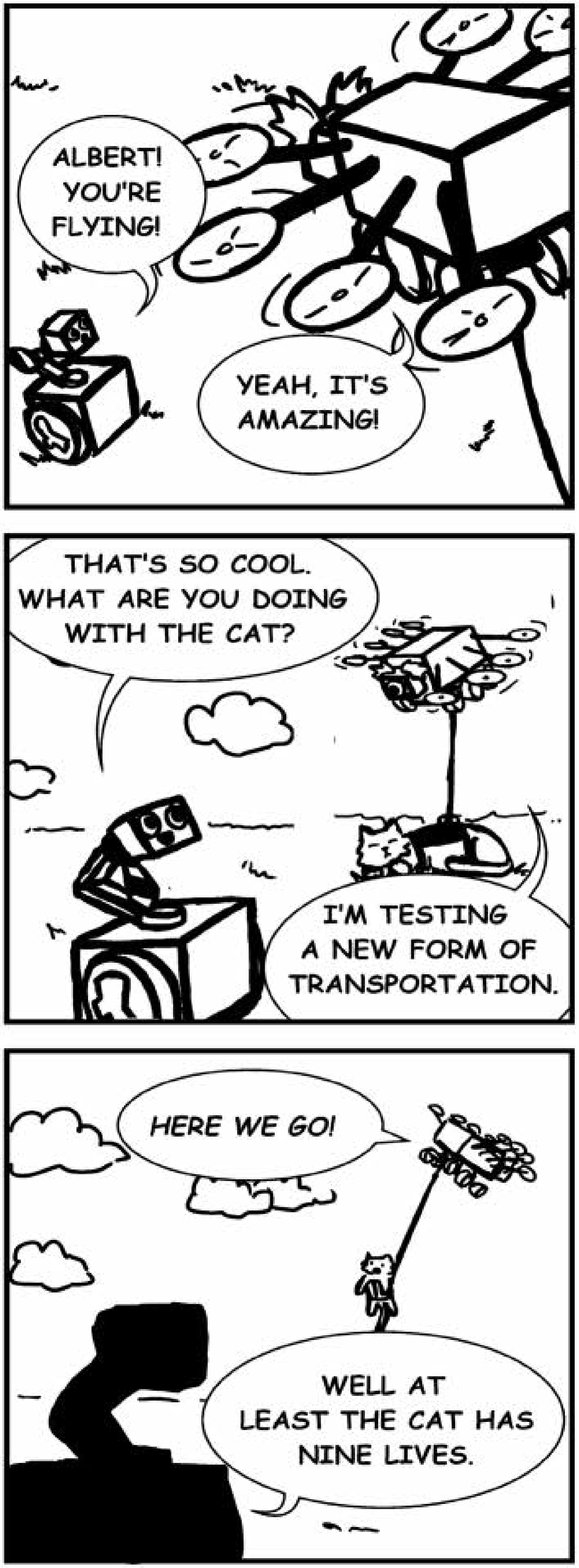
Better yet, all BASIC code and comments to flash the pattern conform to Haiku rules of 5, 7, and 5 syllables. The LED and beeper are wired active low on pin 11 (**https://www.youtube.com/watch?v=WS9e8yj_iLo**):

```
' BS2 Haiku
' Code & comment syllables
' Must form a Haiku

' Also BASIC code
' Must make a Haiku pattern
' flash 5-7-5

Q VAR Byte:PAUSE 1
C VAR Byte:B CON 15
T CON 100

main: HIGH 11
C=B-5
GOSUB flash:PAUSE 1

PAUSE 700
C=B-1
GOSUB flash:PAUSE 1

PAUSE 700
C=B-5
GOSUB flash:PAUSE 1

PAUSE 750
PAUSE 20:PAUSE 1000
PAUSE 10:GOTO main

flash:PAUSE 11
FOR Q=1 TO C
TOGGLE 11

PAUSE T:PAUSE T:NEXT
PAUSE 1300:PAUSE T
PAUSE 7:RETURN
```

That's enough silliness and a wrap for this month. Thanks for all your great questions. Keep 'em coming! Email them to roboto@servomagazine.com and let's see what I can do. **SV**

MAXROBOTECH™

BY: AARON & ROBERT GROSS



CASEY'S CRAZY HEXADECACOPTER

The world's largest homemade drone pulled *Casey Neistat on a snowboard, not just up a mountain, but over buildings and did so with ease. His YouTube video "Human Flying Drone" nicely demonstrates the drone.

The video is so cool that many people thought it was a hoax, so he released another YouTube video titled "World's Largest Homemade Drone." This video shows that the ten foot diameter, 165 pound drone is fully cable of safely carrying a person. He and his rigging added an additional 200 pounds making the total weight lifted about 365 pounds. Redundant systems, 16 electric motors, 16 carbon fiber propellers provide 1050 pounds of thrust, and the motors can draw up to 4000 amps when driven at full throttle.

While I can't recommend a homemade UAV for human flight, the tech is super approachable and allows super capable machines to be built with off the shelf available parts. The uses are endless. While it might be some time before one can use UAVs for commercial uses, the hobby market allows for much experimentation. Just remember to register your drone with the Federal Aviation Administration.

*Casey Neistat, is a successful YouTube video log (vlog) celebrity. In 2015 he developed the video app Beme, and in early 2017 he sold his Beme app for $25 million.

DIY ANIMATRONICS

The Animatronic Body, Inside and Out

By Steve Koci

The process of building an animatronic character starts and ends with a strong body. Unlike us, our characters cannot carefully watch their diet and exercise in order to build a sturdy frame. We must carefully consider the demands we will place upon them, and create structures that are up to the task.

In addition to the primary body construction, we need to consider the costume that our character will wear. This will help us choose the best outer skeleton technique.

There are countless choices when it comes to choosing the method you will employ when constructing your inner and outer body. We will look at some DIY options for both, as well as a couple of commercial systems that will greatly speed up and simplify the construction process.

It is critical that the framework you choose be capable of supporting all of your fancy mechanisms and holding up under the strain of all their wild gyrations. Building a skeletal frame and covering your completed mechanism with an appropriate costume are both critical parts of the build.

Initially, you must determine how much weight and how violent of an action your mechanism will need to support. Knowing this will help you make the decision on your vital support structure.

We will first look at some DIY materials you can employ, and then explore a couple of commercial systems that you may want to consider.

Figure 1. PVC and wood working together.

DIY Framework Options

Building it yourself certainly has its advantages. It saves money and allows you to adapt your design in ways that may not be possible with commercial kits. Plus, the satisfaction that comes with something you have totally built yourself is fantastic. Unfortunately, we do not always have the skills or tools to construct what we need. If you are like me, there never seems to be enough hours in the day to complete my "to do" list. This is where purchasing some of your components may be the right choice.

Perhaps the easiest material to work with is PVC pipe and fittings — just like the materials used for a sprinkler system (**Figure 1**). It's inexpensive, easily found at virtually any hardware store, weatherproof, and simple to work with.

However, it does not provide a very rigid structure. A PVC frame work will tend to move and bend as your mechanism operates. This may possibly work to your benefit as long as the body does not need a lot of strength.

A product that will allow you to have moveable joints with one inch PVC is available from Spider Hill Prop Works (see **Resources**). These will allow you the opportunity to make adjustments in the angle of your joints — a very handy benefit indeed (**Figure 2**)!

Using wood is certainly another viable option. Like PVC, it is easy to obtain at any hardware store, or you may find what you need in your shop woodpile. It is reasonably priced and can be fabricated using tools you may already have on hand. Since it will soak up water, you will want to seal it if it will be exposed to the elements.

Aluminum makes a great body form as well (**Figure 3**). It can be worked with ordinary hand tools and is lightweight.

You also have the ability to manipulate and form it to some extent. The price can add up if you buy it from the

Post comments on this section and find any associated files and/or downloads at
www.servomagazine.com/index.php/magazine/article/March2017_Animatronics_Bodies-Inside-and-Out.

Figure 2. I can move now!

Figure 3. Putting it together with aluminum.

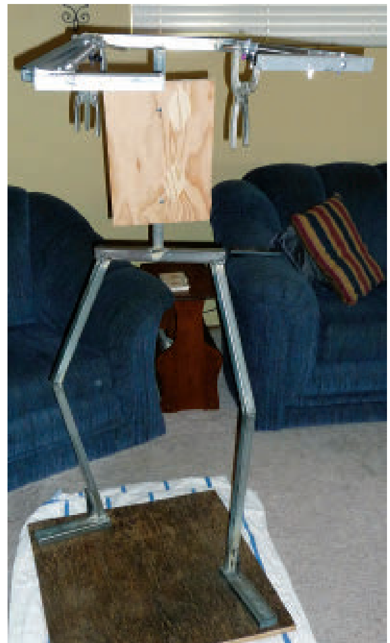


Figure 4. Solid as a rock with welded steel!

big box stores, which is why I purchase most of my aluminum and steel from a dedicated metal supply house.

Steel is the final material we will look at (**Figure 4**). It's not necessary that you have the tools and knowledge to weld as steel skeletons can be bolted together. In fact, I use a combination of welded and bolted joints in order to facilitate disassembly for transport or storage. Bolting your structure together also mitigates one of the biggest shortfalls of welding your steel structure together, which is its permanence.

Figure 5. Combining steel, aluminum, and wood to get the job done.

Once you make your welds, any changes require cutting, grinding, preparing, and rewelding in order to make adjustments. Unlike wood, it will not absorb water but will rust, so be sure to give it a liberal coat of protective paint.

You can always combine materials in a single build. I often will construct the primary frame with steel and then use aluminum for moveable body parts (**Figure 5**).

Prebuilt Kits

There is no arguing the fact that going with a premade system is going to be more expensive than a DIY design, but sometimes the time savings alone makes it worthwhile. The security of knowing that everything will fit together precisely and firmly means there is that much less to worry about.

Figure 6. Designing with Actobotics from ServoCity.

The system I have gravitated towards is Actobotics from ServoCity (**Figure 6**). I use many of the components in my mechanisms, and it makes sense to me to continue that throughout the design.

The buildup is much quicker as I no longer spend countless hours in the shop fabricating my components. My move to these fine products has greatly increased my ability to take my characters to the next level.

Another maker of body kits is AllScare (**Figure 7**). They

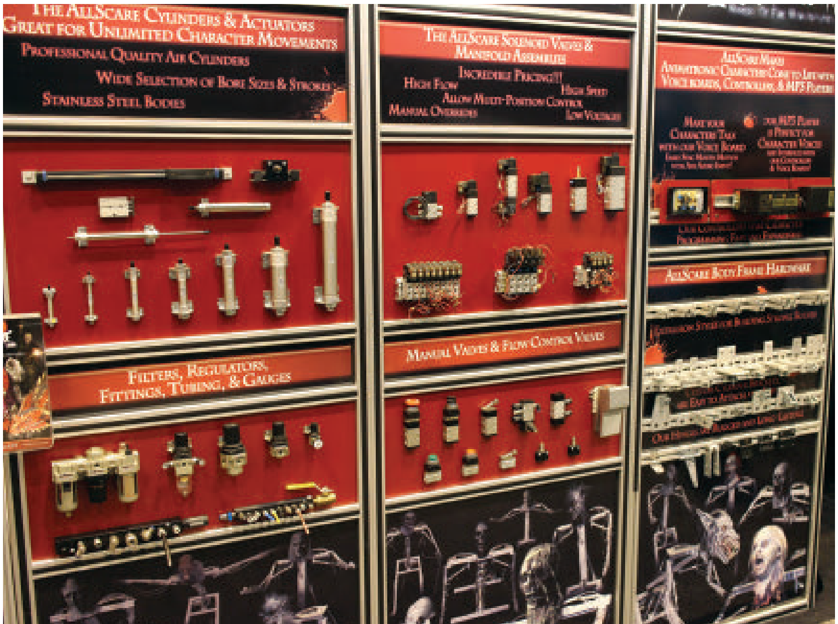


Figure 7. AllScare for pneumatics.


Figure 8. AllScare's adjustable products.

have focused on building products that take advantage of the benefits provided by pneumatics. Their kits are modular and adjustable, giving the builder a wide choice of configurations. It is easy to fine-tune your design by simply loosening a couple of bolts and making your adjustments (**Figure 8**).

Parts (like with Actobotics) are interchangeable which increases the benefits you receive when you make the investment. When you tire of a specific prop, you can disassemble it and repurpose the components into your next project.

In the spirit of full disclosure, I have not personally used the AllScare products. However, I have seen them in action and am impressed with their design. They seem well built and look like they will stand up to heavy use. I hope to have the opportunity to put them to a personal test in the future.

The Outer Skeleton

As with any other new process, we must determine exactly what it is that we are trying to accomplish. My personal requirements are that the body form needs to be easily constructed and provide the correct form in order to allow my character to look natural. It should not require specialized or expensive tools to construct it. It must also be lightweight and inexpensive.

This is a part of the build that I have struggled with in the past. All the mechanical and electronic issues have been resolved, and I am anxious to wrap the project up. It is not simply a matter of dressing our character in the chosen costume. Failure to adequately prepare some sort of body frame is a sure road to disaster! Once your mechanism is completed, you still need to add some external body structure. This not only gives your character the proper shape, but helps to keep clothing out of the mechanisms.

I have tried a variety of methods to achieve the perfect solution, with some methods working better than others. You may find them useful for your projects as they all do have their place. Let me conclude this discussion with my current solution which I am very excited about. It is easy to construct by the DIY builder, is inexpensive, does not take an inordinate amount of time to build, and is fully customizable.

In the past, I primarily relied on shaped aluminum flat stock and foam pool noodles to provide the standoffs for the clothing. I have never been able to achieve the organic look I was searching for. No matter how I formed the aluminum, it still looked like bent metal under the clothing. It was clearly not the look I was after!

I had experimented with other products as well. I started with empty soda and water bottles and then graduated to my wife's plastic food storage containers. If I thought an item had a shape I could use, it was cut up and added to my mechanisms. A tip for all of you non-

RESOURCES

Spider Hill Prop Works — http://tinyurl.com/z6n6wjv
ServoCity Actobotics — http://tinyurl.com/gsyrn53
AllScare — http://tinyurl.com/htkcmxp
Evil Ted — http://tinyurl.com/jd46ox5 & http://tinyurl.com/hfeg6v4
Bill Doran — http://tinyurl.com/j7a5zsg
Magnogrip — http://tinyurl.com/znvhzws
My YouTube channel — http://tinyurl.com/nma2doj
My Website — www.halstaff.com
DIY Animatronics Forum — http://tinyurl.com/qjeehjs

bachelors: Raiding your wife's kitchen for build supplies does not lead to marital bliss!

Chicken wire can be cut and shaped to provide support for your costume (**Figure 9**). It is inexpensive, can be cut with tools you probably already have, and can be easily shaped. However, it is not as rigid as I would like and it cuts me up every time I use it. Operator error I know, but I still do not like it!

I had considered using fiberglass to create a suitable form, but that is a messy and arduous process. Plus, I would still end up with a rigid form with no give to it.

Figure 9. Use chicken wire without the chickens.

Figure 10. We must have the right tools!

Foam to the Rescue

I did an extensive amount of research online in my quest for an improved method for completing my props. I wanted something that did not require a fully staffed production. The Hollywood prop shops do a fantastic job, but I do not have those kinds of resources. I finally had a revelation while searching for costume armor construction details.

The path I decided to take was a technique used by Cosplayers to construct armor and costumes using foam. Foam provides an adequate amount of structure while still being pliable enough to move if something should get out of alignment. It goes to show that you never know where inspiration will come from!

Figure 11. Plenty of foam choices.

Figure 12. My first try at a body form.

Since these will be used primarily as a structure and will be covered in clothing, we will not require the detail that would be necessary if it was used as a finished costume. That makes the fabrication process even quicker!

I must credit Evil Ted and Bill Doran at Punished Props (see **Resources**) for providing the inspiration and video tutorials which helped me on my path to learn this craft.

One of the great things about this system is that it is relatively inexpensive to get started, and you probably have many of the tools already in your toolbox. A sharp razor knife, a steel ruler, a heat gun, a rotary tool, some sandpaper, and a tape measure are the primary tools necessary to get the ball rolling (**Figure 10**).

The foam is readily available from your local big box hardware store, or Harbor Freight which stocks it as floor mats. It also comes in rolls in a thinner thickness. I picked mine up from the sporting goods section at Walmart, but they also carry it at my auto parts store (**Figure 11**).

Having one large piece does allow you to utilize your material more efficiently. The camping pad does have a little bit of texture to both sides but that does not affect its usefulness for my purposes. The square floor mats have a smooth side and a textured side. Choose the style that works best for your application.

I consider my first attempt as a partial success. It is certainly acceptable for my purposes as a body frame, but it leaves something to be desired as a costume piece. I constructed both a female body form and some upper arm sleeves (**Figure 12** and **Figure 13**).

I did learn some valuable lessons. First, cut slowly and

Figure 13. Arm sleeves keep the mechanisms happy.

Figure 14. Barge cement and my homemade glue pot.

Figure 15. Useful DIY sanding sticks.

concentrate on keeping your blade at a 90 degree angle to your work piece. I need to pay more attention to this — especially when cutting curves. Taking your time here will result in much cleaner seams.

Depending on the project design, you may need to dedicate some extra time into pre-forming the foam with your heat gun prior to gluing things together. This will help line things up and reduce the strain on your seam.

Figure 16. One size does not fit all.

I discovered that the mats I had on hand were probably going to result in a more unyielding form than what I wanted. I would like to have it be rigid enough to support my costume, but still have the ability to move and flex if my mechanism should come in contact with it.

I think that the camping mat I purchased from Walmart is a much better solution as it is softer than the floor mats and will provide a more flexible form. One thing I did find as I was putting my first test pieces together with this foam is that it absolutely needs two coats of the Barge® glue cement. It has a more open pore design and it sucks up the first coat.

When applying the cement, do not rush trying to assemble your parts. Allow enough time for it to dry or else you will have a mess. Additionally, both sides are identical, so in order to avoid confusion, mark which side is up to avoid reversing a piece. If you think it sounds like I am speaking from experience, you would be correct. It seems I learn better by making these kinds of mistakes. However, they are frustrating!

There are several useful tools that are simple DIY projects. From Ted's site, I saw how he made a glue pot and some sanding sticks (**Figure 14** and **Figure 15**). I was able to assemble these in a few minutes with things I already had around the shop. (I would appreciate it if no one mentions to my wife their knowledge about the whereabouts of a certain mason jar.) A squirt bottle also works well as a handy glue dispenser.

When experimenting with a new process, I appreciate the opportunity to reduce the startup cost until I am sure it is something that I will continue doing. If I find that I am doing this on a regular basis, I can reassess as to whether I need to invest in the commercial products.

I put the first form together using the plans as they come from Ted's website. As the finished piece was a bit small for my needs, I built a second from the camping mat after enlarging the pattern by 20 percent (**Figure 16**). That may be a bit too large but it was still great practice. This form will be installed on my talking witch from my new scene. You can see that project in the October 2016 issue of this magazine.

I am still a rookie when it comes to working with foam, but I am impressed with the results so far. Like any other

Tools N Tips

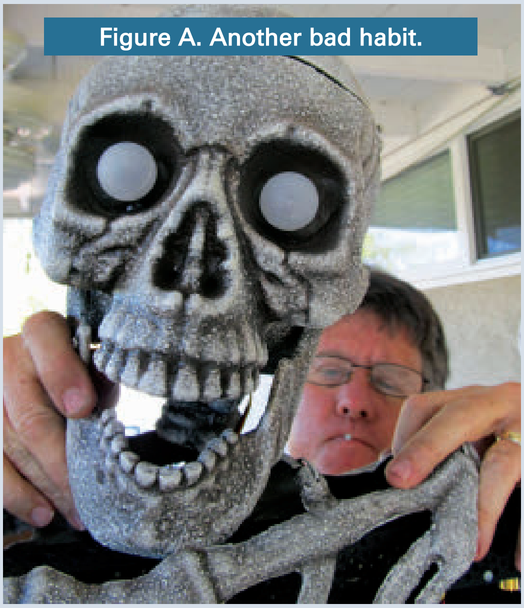


Figure A. Another bad habit.

We will start out this new feature with a gift I received for Christmas. I love getting new tools, and can count on finding at least one under the tree. My photographer son, Bryan — who shoots most of my videos — found a very unique item. When I am assembling a mechanism, I tend to hold the screws in my lips since I do not have an extra hand (**Figure A.**) He has warned that one day I would swallow one. Fortunately, that has never happened and probably will not now, thanks to my newest helping hand.

It's a simple concept, but so handy. It consists of a strap which attaches to your wrist with Velcro™. The magic comes with the two attached magnet strips (**Figure B**). Simply drop the screws onto the magnetized area and they are easily available when you need them. If you are interested, check out the Magnogrip wristband yourself (see **Resources**). They have other magnetic products as well that you may find useful.

This strap — along with my tool pouch — has become a mandatory dress item whenever I am working on a project.


Figure B. No more dropped screws!

technique, my skill will improve with practice. I have a nearly full can of cement and a pile of foam to experiment with. I will be spending considerable time this season sharpening my proficiency with this product.

If you would like to give this process a try, I suggest you watch a couple of Ted and Bill's videos. I have included links in the resources to start you on your way, but make sure to check out the wide selection of videos they have produced.

There is still the issue of how to attach the body form to your mechanism. By using the multi-holed Actobotics channel from ServoCity (see **Resources**) for my internal skeleton, I am able to pin the foam into the perfect position. When adding the foam sleeves to the arms, I will first install a foam pool noodle. I can then attach the sleeves securely to that.

What is That?

I will be adding a new feature starting this month. You may have noticed the "**Tools N Tips**" sidebar. This will be an opportunity to share ideas, tools, and techniques that may not necessarily be relevant to the current article topic. So, make sure to check out each article to see if there is an extra tidbit of information that you may find useful.

If you have an idea that would benefit others, please post it in the DIY Animatronics forum in the Tips and Tricks thread. It may find its way into an upcoming article.

Keep creating, and may the passion to build be with you! **SV**

NEW PRODUCTS

4" Omni Wheel

Mobilize a robot chassis with the Actobotics 4" omni wheel available from ServoCity. Omni wheels can be used just like regular drive wheels but have the advantage of being able to roll freely perpendicular to the drive direction.

The 10 rubber rollers around the circumference of the wheel provide excellent grip to excel in moving in the forward direction. Each roller is mounted on a stainless steel sleeve which rests on a stainless steel axle to allow the rollers to rotate freely and independently from one another. Multiple wheels can be mounted together by installing an 1/8" spacer between the wheels to provide adequate clearance between rollers.

The wheels can be clocked so that the rollers are staggered evenly which smooths out the circumference and increases load-bearing capabilities. The wheels have a 1/2" center hole, and numerous mounting options are provided including the standard 0.77" and 1.50" hub patterns for using with other Actobotics parts.

These wheels can reduce the amount of torque necessary to turn while decreasing the turning radius of your chassis. Price is $9.99.

Ball Bearing Wheel Mount

This wheel adapter is intended to be used with ServoCity's five spoke pneumatic tire/wheel. This piece is machined to perfectly mate with the interior shape of the molded wheel so that the adapter and the wheel will turn as one.

The adapter houses a 1/2" ID ball bearing so that the assembly can be slid onto a fixed shaft and rotate freely on that shaft. The 1.5" Actobotics hub pattern is machined into the adapter so that a sprocket or gear can easily be attached to the outside of the adapter to drive the wheel assembly.

A single wheel adapter can be used and the other side of the wheel can be supported by one of the stock bearings included with the pneumatic tire/wheel or an adapter on each side can be utilized for a more symmetrical setup.

ServoCity recommends running a 1/2" shaft spacer next to the bearing installed into the adapter in order to provide proper clearance between the bearing and the surface that it's running next to. Price is $15.99.

Tapped Wheel Mount

This next wheel adapter from ServoCity is also intended to be used with their five spoke pneumatic tire/wheel. This piece is machined to perfectly mate with the interior shape of the molded wheel so the adapter and the wheel turn as one.

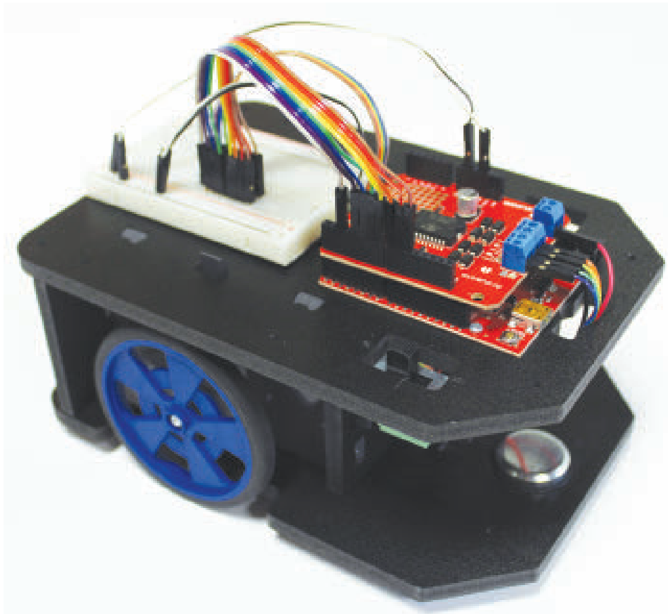
This adapter has the large 1.5" hub pattern with 1/4" thru-holes machined into the face so that ServoCity's 1" HD clamping hub can be attached using 1/4-20 low profile socket head screws.

By installing a clamping hub onto the adapter, the wheel assembly can be secured onto a 1" shaft. This style adapter is commonly used in conjunction with an axle that is mounted in bearings so that the axle and wheel can rotate in unison. Price is $14.99.

For further information on all of these products, please contact:

ServoCity	**www.servocity.com**

Robot Starter Kit

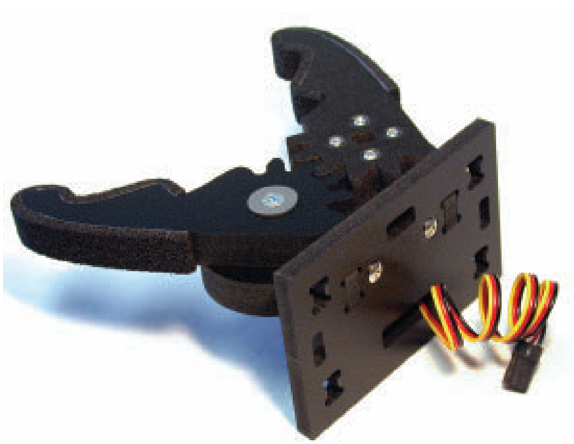


Zagros Robotics has a new version of their Robot Starter Kit — the Gobbit Version 2 — which includes upgraded accessories. This entry level kit includes everything necessary to get started with robots at an affordable price.

The chassis is ruggedly made from both rigid and expanded PVC. It features snap-together tool-less assembly which allows for quick assembly for young roboteers. The chassis has pre-drilled mounting holes for an Arduino controller and a line sensor. Additional components can be easily mounted with double-sided foam or Velcro® tape, or by being screwed into the PVC with wood screws. Simple soldering is required.

The Gobbit Gripper comes with everything needed to add a versatile gear style servo driven gripper to the chassis. Gripper parts, servo, end panel, and battery holder with switch are all included. The kit can be used without the end panel if you decide to add additional parts.

The gripper is made from lightweight 12 mm thick expanded PVC material. The end panel comes pre-drilled to allow for easy positioning and mounting of the gripper to the panel using the supplied #4 screws. On the end panel, the gripper can be easily added to the Gobbit chassis with the jaws of the gripper in either a high or low position.



The MaxSonar-EZ sensor end panel for the Gobbit chassis comes pre-drilled to allow for easy positioning and mounting of the sensor. (Max-Sonar sensor not included.)

The interchangeable end panel is made from lightweight 6 mm thick expanded PVC material.

There's a blank stackable deck for the version 2 Gobbit chassis. The panel comes pre-drilled with a matching six-hole pattern to the chassis' top deck. Use the included threaded standoffs with machine screws to add a single deck. Additional decks can be added with standoffs and long set screws (sold separately).

The deck is made from lightweight 6 mm thick expanded PVC material. Pricing is around $100.

For further information, please contact:

Zagros Robotics	www.zagrosrobotics.com

Continued on page 59

bots IN BRIEF

IT'S ALIVE!

How do you make a robot toy that's both interesting and affordable? It's a tough problem. Making an interesting robot means giving it intelligence and creative autonomy; giving a robot intelligence and creative autonomy is generally not compatible with it also being cheap. However, Bots_Alive — a small company of roboticists — has managed to develop a robotic critter with a carefully thought-out animal-like personality. By hacking an existing robot toy and using your phone as a brain, they're ready to sell it to you for just $35.

Bots_Alive is essentially selling a replacement brain for Hexbug Spider robots. The Hexbug Spider is a robotic toy that's been around forever. It's made out of plastic and costs about $25. It comes with a little infrared controller that lets you drive it forward and backwards and turn left and right. Otherwise, it doesn't really do much.

Bots_Alive turns a Hexbug Spider into a fully autonomous robotic critter by replacing the infrared controller with an IR blaster plugged into your phone and using computer vision to localize a fiducial sticker placed on the robot's head. As long as your phone's camera can see the fiducial, it has full control of the robot's movements, and gives it a life of its own.

Using a phone as the brains of a robot is an idea at least as old as Romo. The advantage of doing so is obvious: We're all carrying around extraordinarily powerful little computers with vision systems and all kinds of other stuff packed into them, so why not steal that hardware and use it for robot control? With this in mind, Bots_Alive picked a platform that has to be one of the cheapest and most common robot toys out there. If you're one of the thousands of people who already has one of the Hexbug Spiders, all you need from Bots_Alive is an IR blaster and their software (which is why the starter kit for Hexbug Spider owners is just $35).

Photo courtesy of Evan Ackerman /IEEE Spectrum.

The kit comes with some blocky things (each with a fiducial on top); the robot will try and snuggle up to the blue ones while avoiding the red ones. It seems straightforward, but the robot will act a bit differently (occasionally, very differently) every time you mess with it. Exactly how those behaviors are programmed and how they come together to give an otherwise brainless robot a personality is where Bots_Alive has a whole bunch of their secret sauce.

Perhaps one of the most refreshing things about Bots_Alive's offering is that it's nothing more (and nothing less) than exactly what it says it is: a little robotic critter.

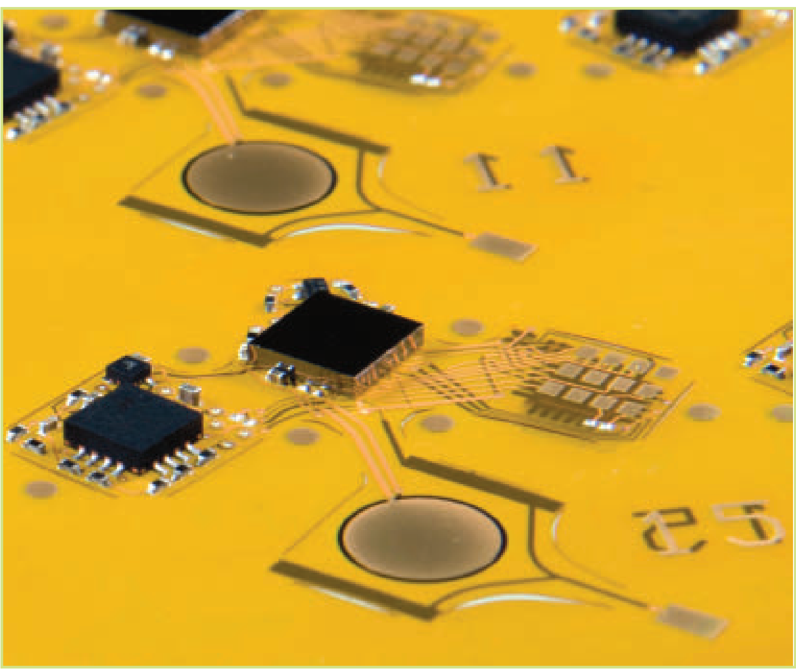
DRAGONFLIES PACKING PAYLOADS

Over the past several years, researchers have managed to steer large insects using electrical implants — a sort of brute-force method with limited real world usefulness.

Now, engineers at the R&D company Draper, based in Cambridge, MA are hoping to overcome those limitations by creating a cybernetic dragonfly that combines "miniaturized navigation, synthetic biology, and neurotechnology." To steer the dragonflies, the Draper engineers are developing a way of genetically modifying the nervous system of the insects so they can respond to pulses of light. Once they get it to work, this approach — known as optogenetic stimulation — could enable dragonflies to carry payloads or conduct surveillance, or even help honey bees become better pollinators.

The DragonflEye project is a collaboration between Draper and the Howard Hughes Medical Institute (HHMI) at Janelia Farm. There are several unique technologies that have been implemented here. The group was able to pack all of the electronics into a tiny "backpack," meaning that small insects (like bees and dragonflies as opposed to large beetles) can fly while wearing it. Some of the size reduction comes from the use of solar panels to harvest energy, minimizing the need for batteries. There's also integrated guidance and navigation systems, so fully autonomous navigation is possible outside of a controlled environment.

Another major advance is that rather than using electrodes to brute-force the muscles of an insect into doing what you want, the Draper engineers are taking a more delicate approach, using what are called optrodes to activate a special type of "steering" neuron with light pulses. These steering neurons act as a bridge between the dragonfly's sensors and its muscles, meaning that accessing them provides a much more reliable form of control over how the insect moves.



Close-up of the electronic control "backpack" before it gets folded and fitted to a dragonfly. Photo courtesy of Draper.

Continued on page 49

COMBAT ZONE

Post comments on this section and find any associated files and/or downloads at the specific article link for each feature.

BUILD REPORT:
C. B. Radio: Garage Door Parts in a Garage Bot

by Aaron Nielsen

Demographically speaking, robot combat is at an interesting point. A growing number of builders have small children in their life who are no longer content to sit placidly on the spectator benches while Daddy and/or Mommy wield hammers, wrenches, and barely-checked profanity to get their bot back in shape after a catastrophic round. They want in on the action. They want to build their own bots — preferably bots bristling with spikes and armed with bazookas.

They (or at least my niece) deflate a bit upon hearing spikes — while utterly deadly in most video games — are considerably less so when one's opponent is a titanium brick. As for bazookas, last I checked, they are almost universally banned, and when they are not they are somewhat challenging to make weight with on a three pound (Beetle) robot.

It was at this point in the conversation my niece found a gargantuan white garage door gear in my junk box and asked if we could build a robot from it.

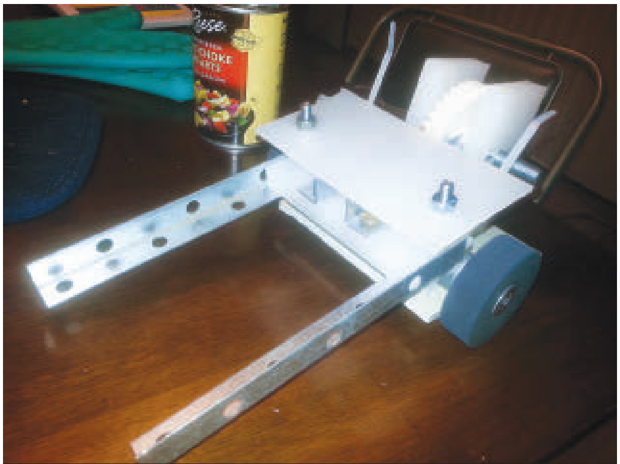
I said, "Why not?" and we went to work on C. B. Radio.

Interestingly, by virtue of her

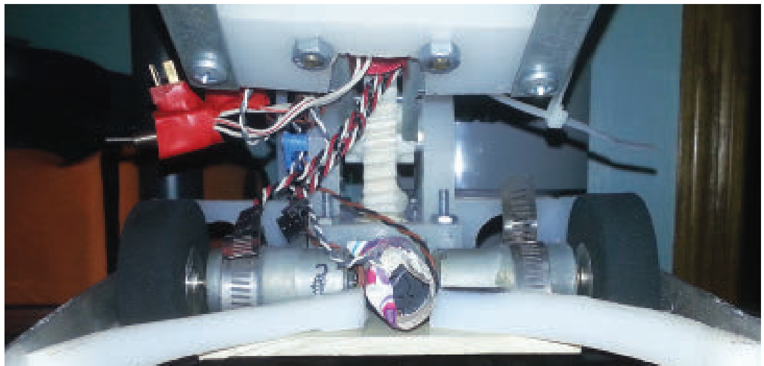
Huge impractical gears? Sign me up!

Featured This Month:

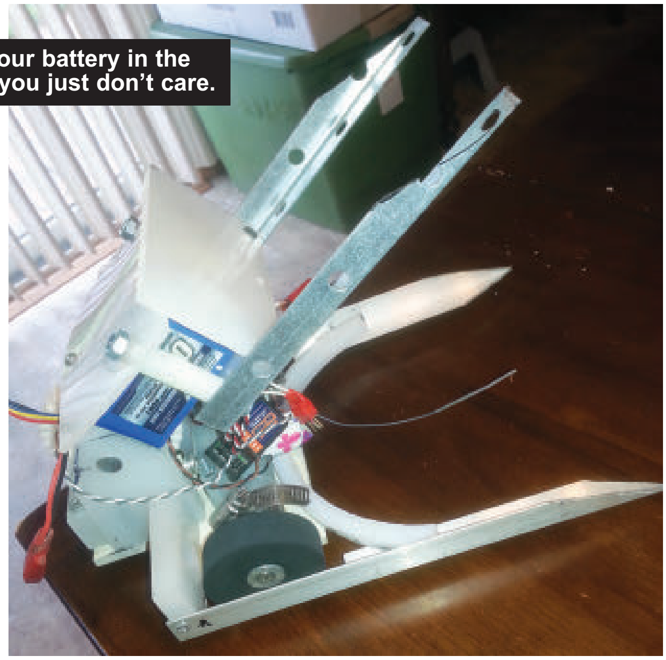
www.servomagazine.com/index.php/magazine/article/March2017_Bot-Garage-Door-Parts.



Early version of the chassis pictured next to a can of artichoke hearts. This may be the first and last time the word "artichoke" appears in an article about robots.



Okay, I admit it. We left the B62 lifter motor (that center thing) exposed like the exhaust port on the Death Star.



Wave your battery in the air like you just don't care.

choice, a number of design decisions had been made for us. Clearly, we wouldn't be playing the "spin metal uncomfortably fast" game. Garage doors are a worm drive with a hard plastic worm and a softer plastic gear. Ask too much of them rotationally speaking, and the worm ceases to be a gear and turns into an auger, reducing your huge plastic gear into a pile of delightful but unhelpful plastic confetti.

Cataclysmic friction loss aside, worm gears do offer one major benefit in the potential for a "shed" load of torque. Thus, the question turns to whether we use that torque to lift or clamp; to which, in the immortal words of Old El Paso taco shells, we said, "Why not both?"

Not being ones for CAD, my niece and I pretty much went straight to cutting — or more accurately, I went straight to cutting. (The bandsaw apparently terrifies nine year olds. In fairness and speaking personally, the bandsaw also terrifies 34 year olds.)

In hacking out our first batch of pieces, we noticed a theme: height. This is not unusual; my garage bots are often crazy tall by the standards of modern bot combat, and with the garage door gear being almost three inches across and the worm adding another inch on top of that, my niece and I looked less like we were building a robot than erecting a monolith.

Humorously, however, we quickly saw we might be able to make our height work to our advantage. There is fairly little 100% agreed upon in bot building circles, but one statement oft uttered and rarely refuted is this: Air makes good armor.

Thus, rather than try to make space on the ground floor of our robot for electronics mounting and material to protect them, we made a little plastic box where the lifting forks emerged and decided to wave our battery, receiver, electronic speed control (ESC), and on/off switch anywhere from three to seven inches in the air, depending on how high we had the forks.

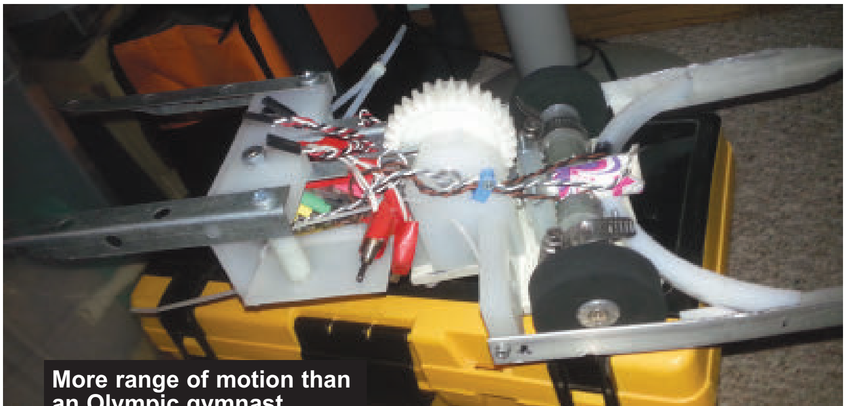
Sure, a spinner could get it, but only if they showed up with a ladder.

One could raise a concern about the weight of the electrical adding to the needed torque for lifting opposing bots, but again, remember these gears are generally used to heft your garage door. Additionally, we powered the gears with a now vintage B62 gear motor.

A B62 is obviously 62:1 in terms of mechanical reduction, and the garage door gears added another 32:1, putting us at somewhere in the neighborhood of 94:1. Sure, unlike an actual garage door, we weren't using a quarter-horse motor to power it, but we still had torque in abundance.

Additionally, worm drive offered another advantage to this lifter/clamper setup. Unlike spur or planetary arrangements, worm drive naturally locks when there is no power being applied to the worm. Instead of having to choose if the lifter/clamper was up or down, we could set it at any height we wanted and be confident it would stay if it impacted a wall or opponent.

Finally, because of the size of the



More range of motion than an Olympic gymnast.

worm gear, we didn't have to machine a fancy shaft mount setup for the lifting forks. Since the gear was so large and would only be making 180 degree rotations, we just bolted the lift prongs directly to the gear itself.

Of course, we also had to stop the robot from tipping over when it tried to do its professional wrestling impression. This is the reason for the static outer forks, which are 1/16 L-channel aluminum and exist entirely to keep the bot from falling forward.

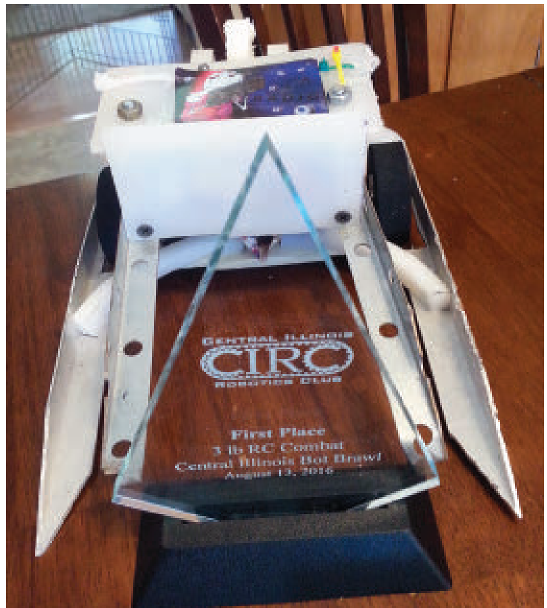
In an ordinary article, this would be the point where the writer would explain about the alloy the forks were made out of and why he/she selected it, but my alloy selection process starts and ends with going to the local hardware store and declaring, "I need some aluminum," followed by, "That looks like aluminum," followed by, "I'll buy some of that aluminum."

As for the other mechanical components, the drive comprised the typical 1,000 RPM eBay discount motors. At fourteen bucks for two, it's hard to go wrong, and I've discovered that instead of purchasing fancy CNC machined mounting plates, I can buy a couple hardware store hose clamps and just sort of clamp them in place.

Also, while the bandsaw might legitimately frighten most nine year olds, they are usually willing to wield a hand drill with wild abandon, but with enough precision to get said hose clamps through the plastic base plate. Our foam wheels were small since — as the lifter could rotate 180 degrees across the bot — we could find a way to (slowly) self-right from any inverted position.

A 11.1 volt 1,000 mAh battery was overkill in the extreme, but it was what we had on hand. Lacking a brushless motor, C. B. Radio only sipped power, and I suspect it could have used a battery as small as 300 mAh and would have only needed charging every other match.

The speed control was a homebrew unit designed to manage up to 30 amps on three channels for a 30 pound robot. In this instance, I was probably asking it to provide less than two amps per channel. It could have been wired with paper clips and still function.



I do have a soft spot for pieces of acrylic with the number "1" on them. Mostly because I see them so seldom.

In closing, my niece was happy with her bot, and I was happy that I had taught her how to drill strategic holes in things, as well as some of the finer points of using off-the-shelf parts for projects other than their intended purposes.

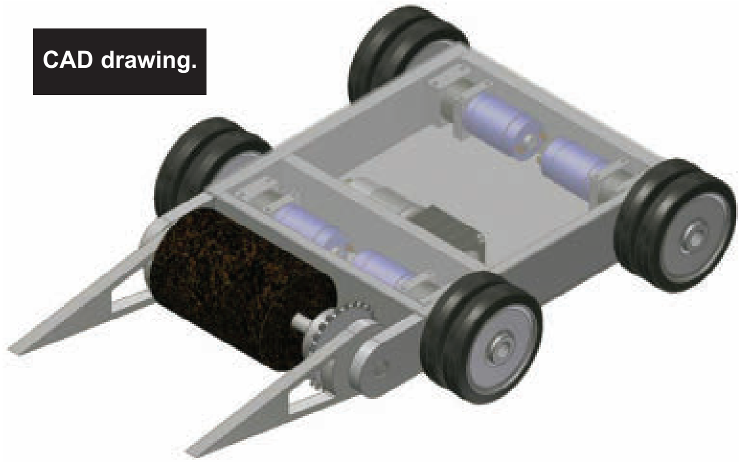
She wasn't able to make it down to the Central Illinois Bot Brawl to drive it in person, but I brought her back a lovely little trophy when C. B. Radio bench pressed its way to a surprising first place. SV

From Nothing to Robot in Two and a Half Weeks

(Or, How to Grow a Mini Yeti of Your Own!)

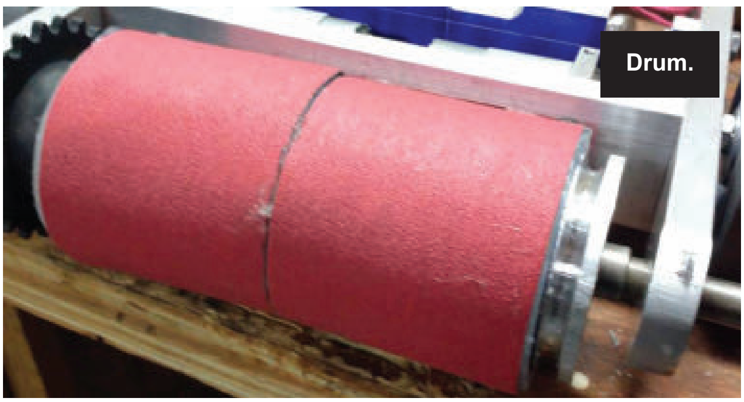
● by Alex Horne

In the depths of August 2016, I attempted to order brushless motors and speed controllers to convert my Featherweight (30 lb) Also Riptoff to brushless drive for power and weight savings. My favorite low cost/lower quality vendor on the Internet (AliExpress) typically takes 1-2 months for things to cross the ocean, and does not believe in tracking the small Vietnamese fishing boats holding your packages. Having them www.servomagazine.com/index.php/magazine/article/March2017_Robot-Build-Two-Weeks.



CAD drawing.

Motor.



Drum.

show up at your door at all is always a welcome surprise, like some kind of Birthday Roulette. If the thing you bought from AliExpress is funny — like motivational mini traffic cones or an inflatable party cactus — this is great as you get a surprise funny thing you probably forgot you bought. If it's a robot part and you have about three weeks before your next event, this creates a classic 'uh-oh spaghetti-o' rage-build situation.

Quick thinking and several visits to Harbor Freight gave me three options:

#1: Drop about $200 I didn't have on electronics and motors to get what I ordered from AliExpress state-side.
#2: Play Swiss cheese with a power drill to get

Chips.

Parts.

Also Riptoff's frame on a diet.
#3: Build a whole new robot from scratch.

I had been having dreams of making my own little mini Yeti (that awesome drum spinner/lifter robot from BattleBots™), and nightmares of a terrible Sportsman robot that used a sander to ruin paint jobs. To appease my therapist and my seven fans waiting for my next silly robot, I combined those two ideas and built a Sportsman mini Yeti, substituting an angry toothed spinner for a drum sander to really grind down my opponents. This is real Yeti.

With the shipping box locked and the lights on, it was robot-building time. I knew it must look like a smaller Yeti and shipping took more time than there was to Franklin, so I could only use parts I already had (excluding sand paper).

With these incredibly strict criteria laid out, I started hunting in the robot graveyard hiding behind Wedge Industries HQ. I found four "wangboxes" (an obscure Chinese gearbox), a Versa gearbox, and a stable full of HobbyKing Donkeys (a scary cheap motor, even for HobbyKing).

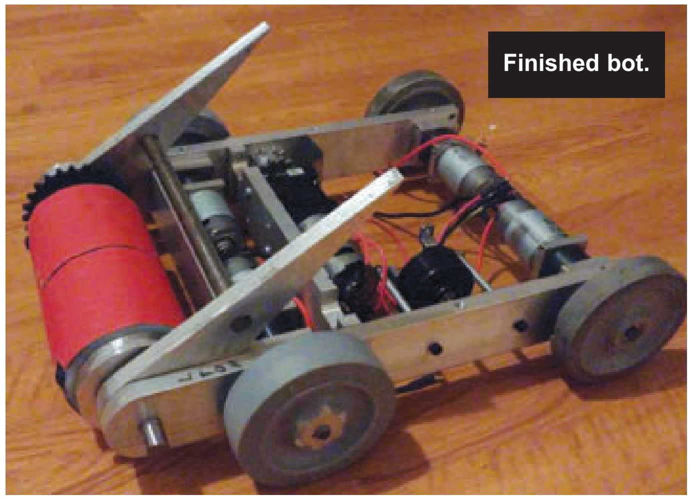
It was then time to hook up the dial-up modem and jack into the finest of virtual simulation environments: SolidWorks.

By importing pictures of our furry friend from


More chips.


Gear.


Shiny bar.


Finished bot.


Yeti.

True Grit.

the frozen oil fields, I was able to directly plagiarize all of their important dimensions and scale it down to something far more travel-friendly. All of the electronics I had on hand were pretty generic, with one BotBitz 30a per drive motor, a 70a HobbyKing Redbrick on the sander drum, and a Holmes Hobbies BR-XL on the lifter.

With all this powered by a handmade fire-proof 5s A123 pack, it was only a matter of playing Tetris with parts and making the bits that I just couldn't buy.

Detail is for chumps and people who don't have a machine shop. I've always found it's easier to create pretty CAD models by just making the real thing and using the right amount of fly-cuts and fuzzy backgrounds. Besides, it's more fun to have a rough design and try to fill in the details as you go. (Most of the time.)

Unfortunately, I spent enough time in cyber-space such that it permanently damaged my brain and I had no choice but to venture forward into the machine shop, only communicating through grunts and photographs.

I like finding excuses to use a rotary table. I settled on 1/2" x 2" aluminum for the frame and lifting forks because I had enough of it to make the robot. The top and bottom are 1/8" polycarb.

The Versabox has a 1/2" hex shaft. The sprocket originally had a 1/2" bore. I don't own a 1/2" hex broach. I used a 1/8" endmill and a Hartford head (indexable vise) to make a hex bore.

I put a 1/4" radius and flycut the back piece because anything worth doing is well worth doing right.

The drum itself was a random

piece of plastic that I think might have been nylon, or maybe Delrin, or possibly UHMW. Aluminum endcaps kept it from flexing and 1/2" needle roller bearings kept it rolling.

I would have loved to do a solid aluminum drum, but that would have cost money in parts I didn't have. For maximum abrasion, sand paper was originally 120 grit but I changed that to 40 grit by the time of the event just to rub people the wrong way.

Notice the different size wheels? It would have driven better with four wider wheels but sacrifices needed to be made for those sweet Yeti aesthetics. I was now finished and Yeti to fight.

True Grit's design was a last minute decision based on a desire to ruin people's paint, have some fun, and make something different than my usual single tooth vertical spinner.

Even though it went 1-2 at The Battle On the Parkway, I still call it a success. 8/10 would be True Grit again. SV

New Year's Resolution — Part 1

● by Pete Smith

My 12 lb (Hobbyweight) combat bot, Isotelus Rex (**Figure 1**) did not fare well at the Motorama event last year. Its lightweight titanium wedge was torn up (**Figure 2**) in a fight with Minor Threat 2 (**Figure 3**). It was now clear it was no longer a match for the new generation of vertical spinners.

It had plenty of speed and power, and the chassis had proven itself to be tough enough, but there was not enough weight allowance for a really effective wedge. Isotelus had to go on a weight loss program!

First thing I looked at was the drive. I already was using a brushless drivetrain using 2858 1,200 kv Tacon motors with 16:1 BaneBots P60 gearboxes; this had more than enough power. Perhaps a smaller lighter motor could be found?

A search turned up the similar but shorter Tacon 2838-380-30T 2,200 kv motor that weighed 3 oz less each. Not a huge amount, but a start. The higher kv was a concern as the existing motor was already fast enough at 4S liPo, but I ordered three and fitted them to the existing 16:1 gearboxes.

On taking the old motors off, I noticed that the motor spur gear had only been partially engaged (**Figure 4**) with the first stage gears on the gearbox. This hadn't caused a problem during the short time it had been used, but was obviously not ideal. The shafts on the new motors were a little longer and I pressed the gear on with my bench vise (adding a little Loctite 640 on the shaft), leaving it a little further out (**Figure 5**) to ensure full engagement. I checked the weight (**Figure 6**) and confirmed a savings of just under 3 oz per assembly.

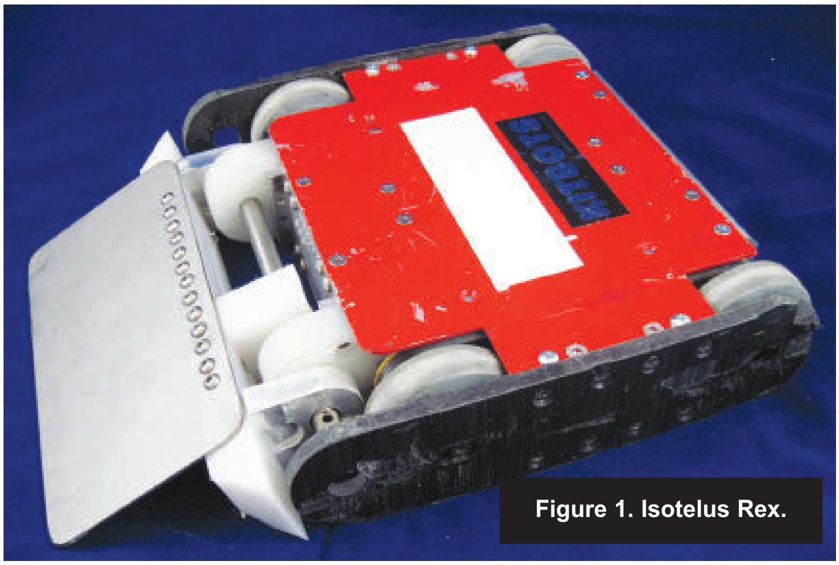


Figure 1. Isotelus Rex.

I installed the new assemblies in the existing chassis (**Figure 7**) and tested out its performance. It was fast

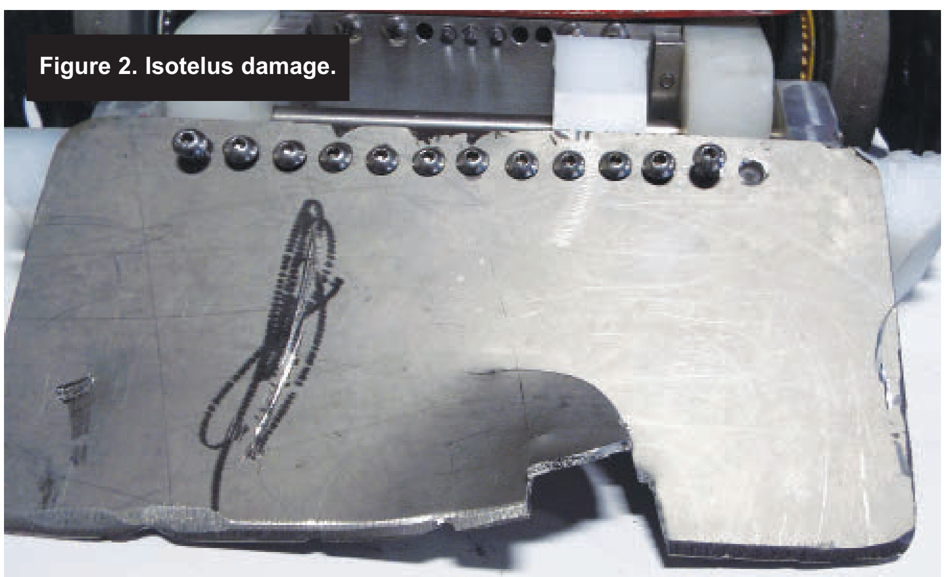


Figure 2. Isotelus damage.

www.servomagazine.com/index.php/magazine/article/March2017_Robot-Rebuild-Isotelus-Rex.

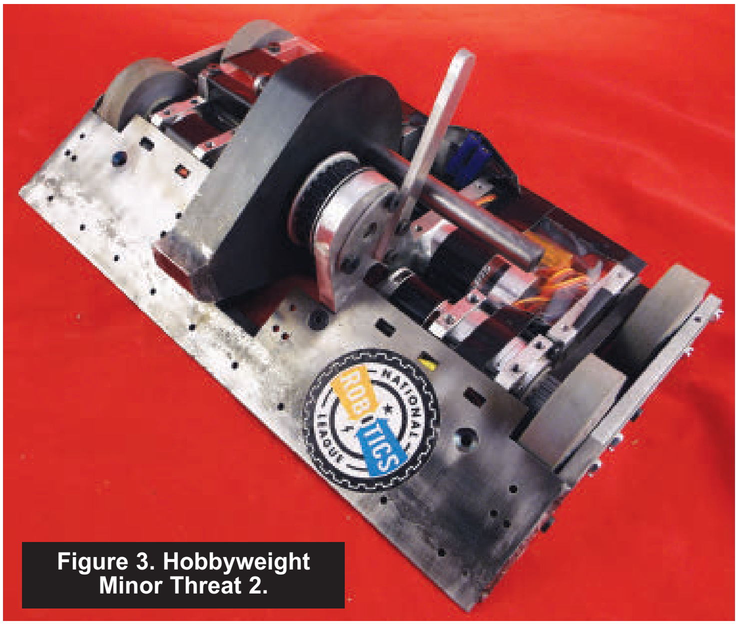


Figure 3. Hobbyweight Minor Threat 2.

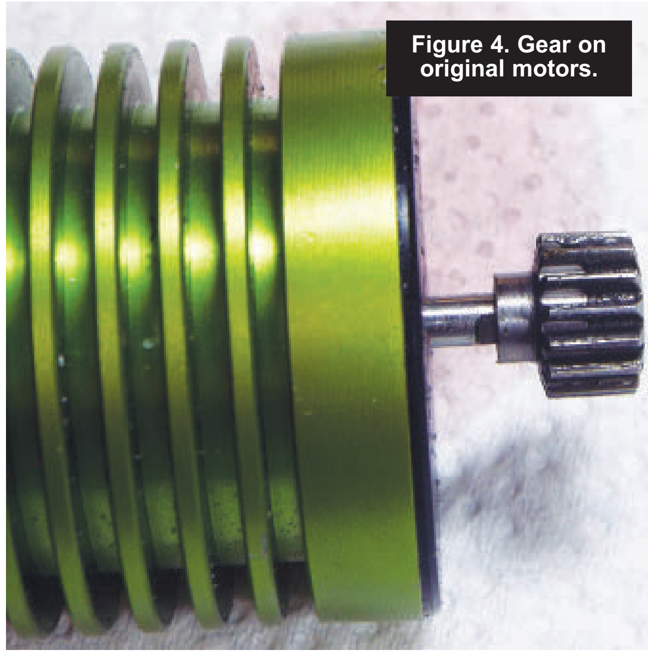


Figure 4. Gear on original motors.

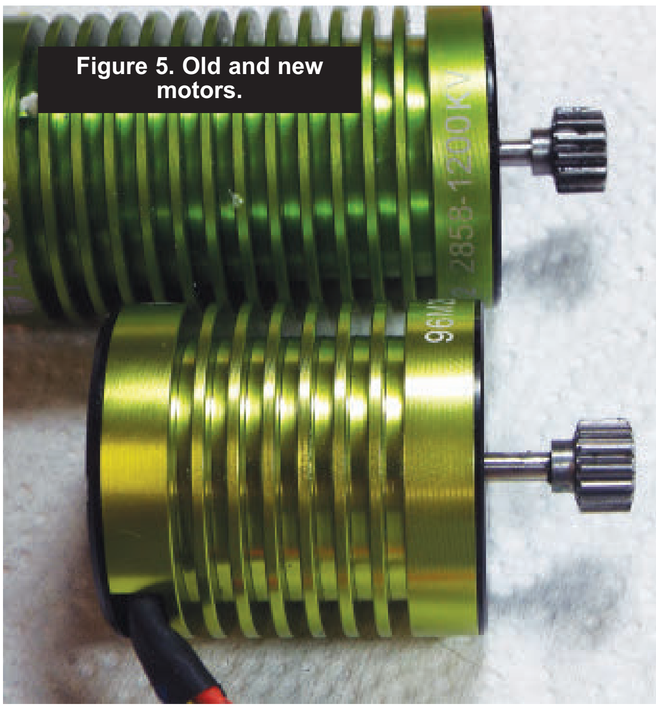


Figure 5. Old and new motors.

Figure 6. Confirmed weight savings.

— very fast — due to the higher kv of the motors, even when loaded up to about 20 lbs (to simulate carrying another bot on top of it). It still had good acceleration and would spin its wheels rather than stall if pushing up against an immovable object. I ordered some 26:1 gearboxes to reduce speed to a more manageable level, with the added bonus of increasing torque at the same time.

So that was 6 oz freed up for a better wedge. I thought about using smaller batteries and lighter ESCs (electronic speed controllers) which might help with another ounce or two, but that wasn't going to get me where I wanted to be.

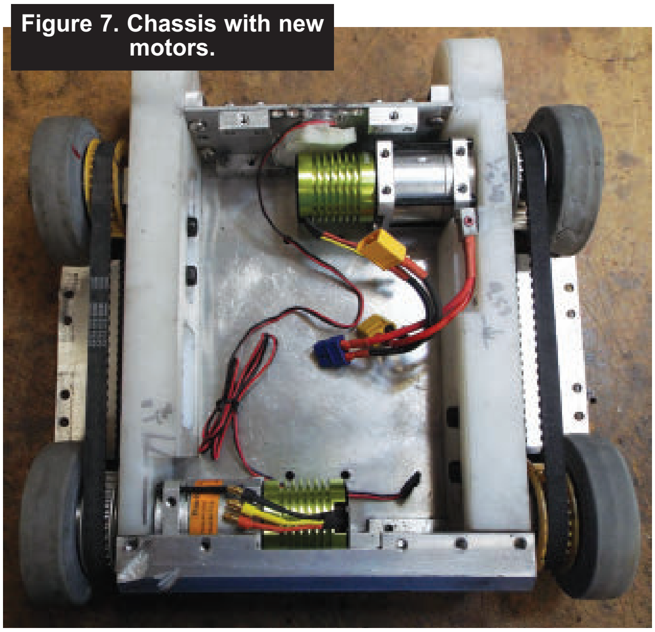
The original dimensions of the bot had been decided by the Speed 700 brushed motors used in the first version, and it was clear that the bot was now much wider than it needed to be with the new much shorter motors. A quick measurement revealed I could take a full 2" out of the width. This would save a little in the chassis but would allow the wedge to be a full 2" narrower as well, allowing a thicker wedge in the same weight allowance.

The narrower wheel base should also help somewhat in driving in a straight line.

The original CAD was quickly modified (**Figure 8**) and new front, top, and bottom panels were ordered.

The build-up will be covered next month in Part 2. **SV**

Figure 7. Chassis with new motors.



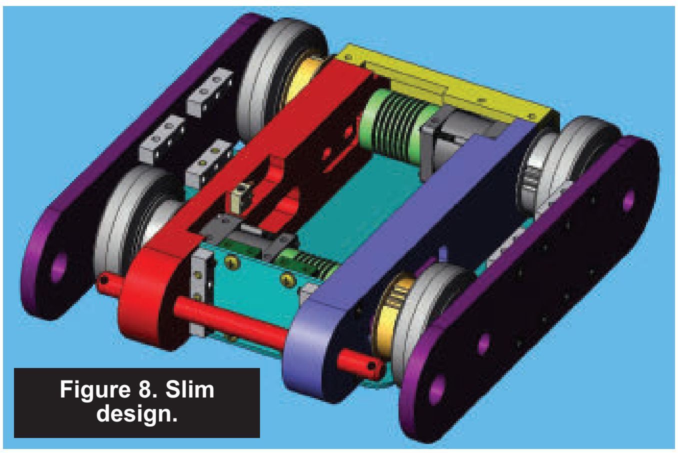


Figure 8. Slim design.

RoboGames in Retrospect

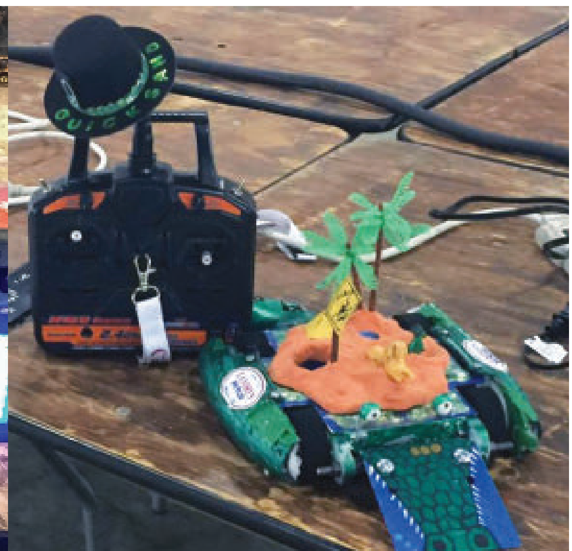
By Mark Elam

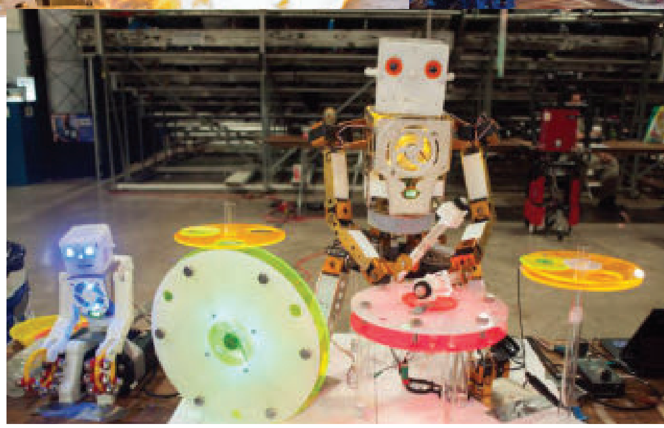
Post comments on this article at **www.servomagazine.com/index.php/magazine/article/March2017_RoboGames.**

The crowd lights up. What was once a lethal 120 pound killing machine now lies lifeless on the arena floor; its batteries producing a white-hot flame that jumps from the gaps of its shell. The referee looks back and forth at the two members of the safety crew before declaring enough is enough as the buzz of the audience fills the room. Charging in, extinguishers blazing, the fire is quickly put out. The opposing robot is deactivated as quickly as its driver can charge through the growing veil of toxic smoke. With thunderous applause, the charred cadaver of the middleweight is hauled outside as the arena is left to air out.

It's only my first day on the job, and chaos is already breaking out. Just as planned.

Like every member of the robot combat community, I know about RoboGames. David Calkins, president of the Robotics Society of America, started the event in 2004 as ROBOlympics, and it has only grown bigger since then. Gone are the days of frequency crystals and NiCads — nostalgia does not bring one closer to the coveted gold medals that have been the top prize in the sport for over a decade. Most machines have ditched their aging ETeks and





Vantecs for high power brushless motors and the latest in speed controller technology: the Ragebridge 2 or Victor BB.

But not everyone.

In the interlude, I walk through the pits, observing the plethora of bots in all weight classes being readied by their crew members to fight on a moment's notice. Kelly Smith works to fit a new leading edge onto Psychotron's wedge — it's the first time the robot has fought at competition since 2004. So too, is the case for Knucklehead, the lightweight who ended up being knocked out of the competition in the first round, as it had been back in 2002. On the far side, Original Sin — a multi-time champion and one of the bots favored to win — is getting some TLC from Gary Gin, whose strategy of powerful four-wheeled wedges has netted him multiple gold and silver medals over the years — even a ComBots Cup.

Last Rites — the perennial machine of death and total destruction — sits nearest the arena as Ray Billings and Rick Russ work on the former. Considering that a decade ago, their machines were a thwackbot and a heavyweight with 'active armor,' it becomes ever more amazing to gawk at the beautiful design of Last Rites' spinning weapon. So too, is Swamp Thing — the durable spinner killer sporting its front wedge with two steel additions; in time, these too will be chewed up by the infamous Brazilian machine.

The Brazilians stayed together. Team ThundeRatz, Team U!airrior, and the infamous Team Riobotz — all work frantically on their bots, punctually stating their needs for parts and tools in Portuguese to one another. The love of the sport and camaraderie showcased by them — teams who have regularly battled one another in Brazil for years — almost appears to make Calkins smile as he walks by. Nope, must've just been imagining things.

For those of you who don't know anything about RoboGames, it's the largest open robot competition in the world. A massive variety of mechanical disciplines are practiced over a period of three days at the event, which brings forth the absolute best competition from around the world. All desire to do the same thing teams do at the Olympics: to bring home a medal and gain further glory for their country in the electronic arts.

Ten categories generally detail the sorts of challenges found at the event. While RoboGames is most known for being the home of extremely brutal robot combat in eight weight classes, it does have far more going on than that. Humanoid robots compete against one another in kung-fu, or Sumo — sometimes they get fitted with airsoft guns and shoot at one another. A wide variety of more traditional tracked and wheeled Sumo classes are found as well, in classes ranging from 25 grams to three kilograms. Bot-hockey is a popular event, and a wide variety of artbots (some of which mix and pass out drinks) often enamor the crowd, dazzling them with spectacular displays of light, sound, and action.

Kids — being an integral part of the community — have always been welcomed at RoboGames, with the Junior League existing especially for them. For most, it wouldn't be fair for them to tangle with professionals with decades of experience straight away. Most is a relative term, however. During the 2016 event, I was informed that three kids — all under 13 years old — had swept the adult three pound combat category, taking gold, silver, and bronze and leaving the 40 year old men to sadly go home medal-less.

Saturday grew more ecstatic as the tournament proceeded. T-800 sent itself flying around the arena as though it were channeling the essence of Tornado Mer,

balancing on its side as it rolled across the arena, trying to right itself and bash the Brazilian bot, Apolkalipse into immobility.

Vlad the Impaler II is back again, as well. The creation of Gage Cauchois — a veteran of the original robot combat events — has seen much action, passed between builders. This year, the Hjeldens are driving as it faces off against Robot Wars competitor, Foxic. The last time Vlad II fought a British robot, it suffered a humiliating defeat. However, this time, it's the orange machine that suffers a breakdown, leaving Vlad's luck pointing upwards.

Craig and Chris Danby — both of Team Danby — are the only British team competing in the combat robot weight class. Their robot Foxic still bears the scars of its appearance on the BBC — and of Last Rites, whom it took excessively well. Craig dislikes the design, and indeed that Foxic would find itself completely replaced with a new machine, readied for next year's event. The almost 20 year veterans of the sport intend to win the 2017 gold medal.

The Brazilians may not have worked together, but they were exceptionally successful. Touro Maximus' win over Original Sin in the grand finals of RoboGames 2015 seems to have paved the way for Brazilian teams to keep winning. More medals were won by Brazilian robots in the three large weight classes than by American robots — the first time this has ever happened. Apolkalipse, Touro Classic, and Touro ensured that no American middleweight would win a medal, while Uairrior's Federal MT claimed the gold from the American ProPain-clone Tastes Like Burning in the grand finale. Touro Maximus was not up to snuff, only taking a surprising third as Original Sin regained its crown for the first time since 2013.

As time passes, registration opens up for this year, and registration is fierce. Half a dozen Brazilian teams intend to replay the Brazilian lightweight championship in California. Herr Gepounden makes a return, slipping into the last slot as lightweights completely fill up in record speed. The heavyweights soon follow, with robots like Sewer Snake and Megabyte making their return to the Combox. Team Echo signed up; though they lost to medalists last year, their performance was exceptional. Collegiate teams from the University of Virginia, University of Texas at Arlington, UC Merced, UCLA, and Georgia Tech all have sent teams to compete. School teams have been regulars at RoboGames. Universities and high schools have often found combat robotics to be a fun way of teaching engineering to students.

RoboGames 2017 will be an event to remember. With record numbers of registered robots, the highest quality competition in the world over multiple weight classes and events, and teams eagerly awaiting the contest this April, something special is certainly coming.

Don't miss it! **SV**

Photos graciously provided by Jon Bennett, Michael "Fuzzy" Mauldin, Sávio Mendes, Ian Miller, and Team Ocean Robotics.

The Dobot Magician

A Robot Arm Accurate Enough for 3D Printing

Those interested in a quality robot arm or a 3D printer should consider the Dobot Magician. Its accuracy and easy-to-change end effectors give it multiple personalities that can meet a variety of needs.

By John Blankenship

My life-sized robot, Arlo was featured in the January 2015 issue of *SERVO*. Arlo had two inexpensive servomotor powered arms that allowed me to experiment with using vision and touch sensors to control the arm's movements. Like most arms built for hobbyists, Arlo's limbs have significant limitations when it comes to accuracy, repeatability, and payload weight. For simple experimentation, such limitations can be acceptable, but real world applications that perform useful tasks often require better specifications.

Unfortunately, most arms capable of dealing with realistic problems are designed for manufacturing environments, which means they are priced far beyond a hobbyist's budget. I enjoyed experimenting with Arlo's arms, but I wanted more capability. So, I began looking for an arm with decent specs and a reasonable price.

Post comments on this section and find any associated files and/or downloads at **www.servomagazine.com/index.php/magazine/article/March2017_Dobot-Magician.**

Figure 1.

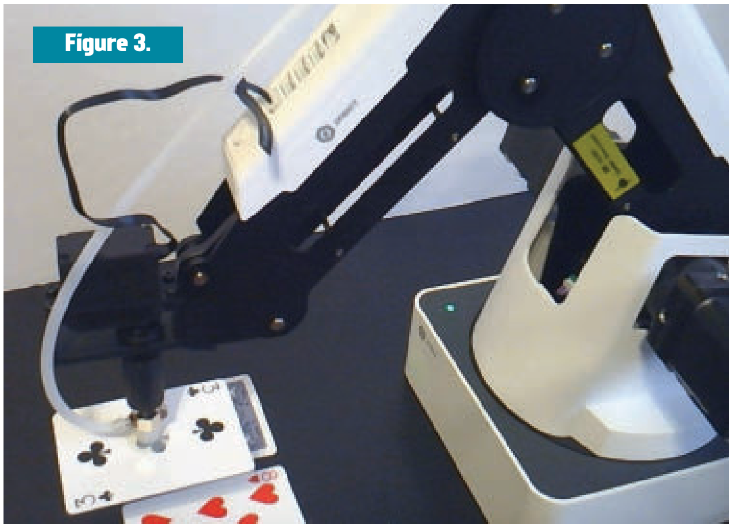


Figure 3.

I was somewhat discouraged until I found the Dobot Magician robotic arm, which provides far more options and capabilities than I had even hoped to find. **Figure 1** shows the arm and an impressive assortment of accessories as I unpacked everything.

It has four degrees of freedom, with most of the joints powered by stepper motors to obtain an accuracy of .2 mm when handling loads up to 500 grams.

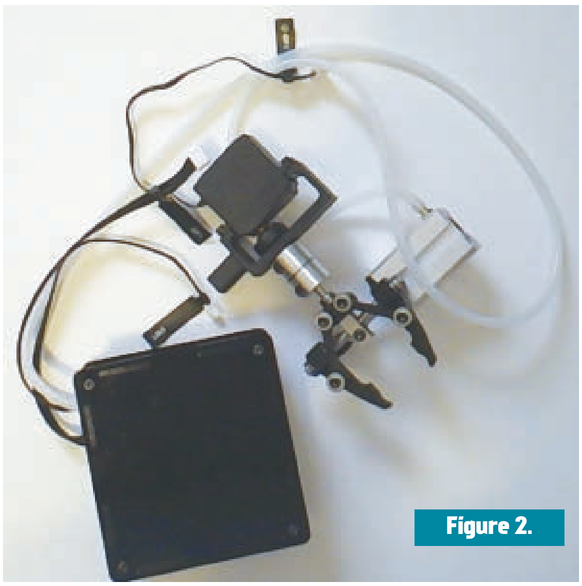


Figure 2.

End Effector Tools

One of my favorite features of the arm is how easy it is to change the tools used for interacting with the robot's environment. A few turns of a single butterfly bolt quickly releases one tool and attaches another. I am already starting to think about building a custom apparatus for attaching a Dremel tool for drilling printed circuit boards (PCBs).

The Magician comes with two *hands* for picking up objects. One resembles a typical gripper as shown in **Figure 2**. The fingers always point downward, and open and close pneumatically (an air pump is included). The second option for picking up objects is a suction cup (also powered by the air pump) that allows you to pick up small blocks or even objects such as playing cards (**Figure 3**).

The ability to pick up cards was particularly exciting for me, since one of my many hobbies is magic. I am already envisioning my Magician living up to its name by performing card tricks for my friends.

I/O Ports

Figure 3 also shows another well-thought-out feature of the arm. Notice the sockets on the robot's forearm (the protective covers have been removed from all sockets). They provide both digital and analog ports as well as switched power and PWM that can be used to control end effector tools and obtain sensory information. Additional ports are available on the arm's base for interfacing with devices not mounted on the end of the arm.

Since some form of gripper will generally be the primary tool used by most hobbyists, I will explore Dobot's software for controlling the arm and grippers shortly, but first let's look at some of the Magician's other end effector tools.

Drawing and Engraving

There is a spring loaded mechanism for holding a pen or even a paint brush to give the arm the ability to draw or paint. There is also a laser capable of engraving (burning) gray-scale pictures on heavy paper, wood, or leather. I was impressed when I discovered how the provided software makes it easy to perform these tasks. With engraving, for example, you just give it a picture (or text) and it handles everything else. I have hopes that such ease-of-use may even get my wife (an artist) interested in robotics.

Figure 4 shows the pen holder and the laser. **Figure 5** shows how accurate and repeatable the arm is. The smiley face shown was draw 10 times with the pen attachment.

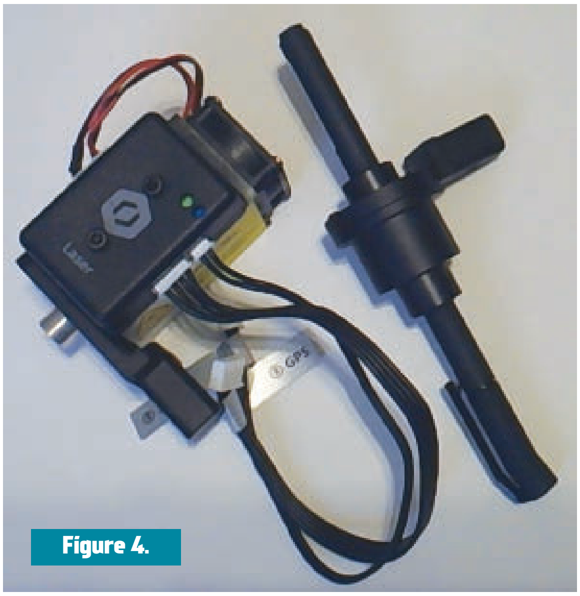


Figure 4.

As you can see, the nine subsequent drawings coincide almost perfectly with the original image.

It is worth mentioning that most operations — such as engraving and moving objects — perform their actions without external friction restricting the arm's movement.

When writing with the pen, the pen-to-paper contact can source considerable friction though, especially if the arm is initialized with too much force on the pen. Minimizing such friction is vital to getting the performance shown in **Figure 5**.

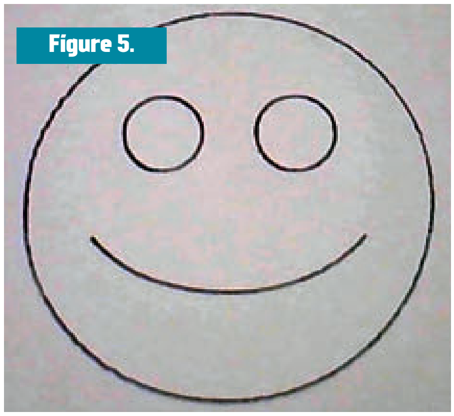


Figure 5.

Figure 7.

3D Printing

The arm's accuracy makes it possible to use it as a 3D printer. This dual functionality has the potential for making the arm very affordable for hobbyists thinking of purchasing both an arm and a printer. All the necessary components including the software, a temperature controlled print head, and an extruder (see **Figure 6**) comes with it. Even though I had never used a 3D printer, I was able to print the thin-walled twisted vase (holding a tube of Super Glue™) shown in **Figure 7**.

After learning more about printing, I was able to produce the small squirrel-like Pokémon character also shown in **Figure 7**. It would be beneficial if Dobot included a 3D tutorial for beginners. I am sure that I could benefit from a better understanding of how various printer parameters affect the final product. Even simple hints such as *"minimize friction in the extruder conduit by keeping it as straight as possible"* would have been nice to know without a lot of trial and error.

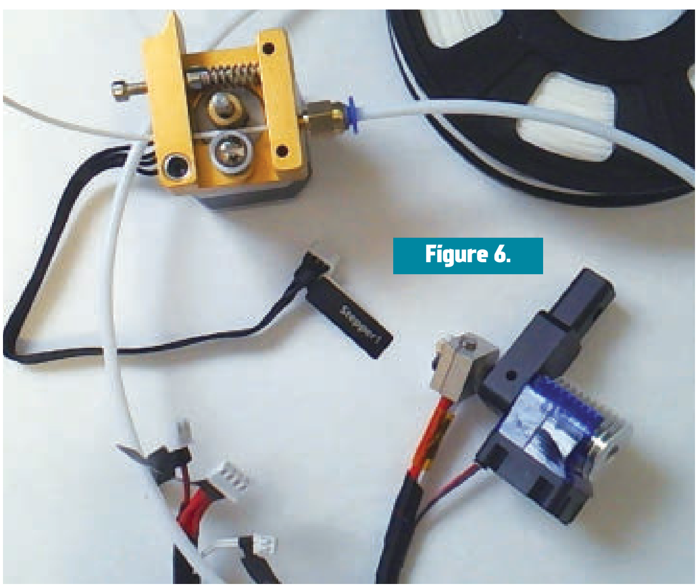


Figure 6.

The Dobot Studio

All of the arm's functions can be initiated from an integrated environment called the Dobot Studio as shown in **Figure 8**. Being able to navigate easily to various functions is a nice touch.

Programming the Arm

I expect that most hobbyists will want to program the arm to perform interesting applications. Imagine, for example, having the arm actually moving the pieces when playing checkers, or even drawing the grid for a game of Xs and Os and then marking its own moves.

The arm can be programmed with an integrated scripting language or with direct commands through a serial interface or over a network connection. Dobot provides the basic protocol for such direct commands, but there is an even easier method. A variation of Google's Blockly language is also included, and is by far the easiest way to customize the arm's behavior (although complex applications *might* be better implemented in one of the more robust programming alternatives).

Figure 9 shows a simple Blockly program that demonstrates how easy programming the arm can be.

Some of the details for programming the arm are currently lacking due to its newness, but Dobot promises to have complete programming manuals available by April 2017. I had a few installation problems and numerous questions about programming the arm, and was pleased with the help obtained from Dobot. Patience will be a virtue though, because the time difference between China and the United States almost ensures that answers to questions will take at least a day or two.

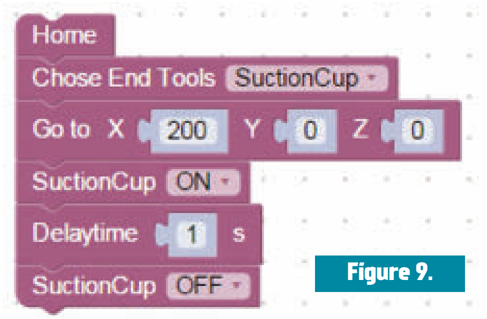


Figure 9.

Figure 8.

Teach and Playback

The ability to truly program the arm will be appreciated by most hobbyists, but for many applications, you can simply move the arm to various points and record the joint positions in a table (see **Figure 10**). The arm can be moved electronically using the control panel shown in **Figure 11**, or you can disengage the motors by pressing a button on the forearm to manually place the arm in the desired positions.

Notice there are two sets of controls in **Figure 11**. The lower two move each joint independent of the others. The upper two controls allow you to move in the X-Y-Z directions without affecting the orientation of the gripper. This requires many calculations, and I found it impressive that the arm handled them all for me.

Once a table of points has been created (and edited with the integrated editor) the arm can be instructed to move between the points in a variety of ways. It can, for example, move between two points using a straight line, or you can use the JUMP mode to have the system *automatically* raise and lower the arm when moving between two points (a behavior perfect for manipulating objects with the suction cup or gripper).

Entries in the table can force the arm to delay for a specific period of time before moving to the next point, or even pause until the value of a port (either analog or digital) returns a specified value. You can also tell the arm to repeat the table actions a specific number of times.

Once you have developed a table to solve a particular application (drilling a PCB, for example), you can download everything to the arm with the click of a mouse. The arm can then be disconnected from the PC and perform the operations on its own.

If you only want a 3D printer, the Magician is probably not a good choice because there is so much competition. At the time of this writing, even Walmart.com offers six 3D printers with three of them selling for under $400. If you have been thinking about purchasing a quality robotic arm though, and you want/need something better than a typical servomotor powered hobby arm, then you should check out the Dobot Magician. It's like buying a great arm and getting the printer for free. Even with a price tag around $1,000, its many tools and features make it worthy of consideration. **SV**

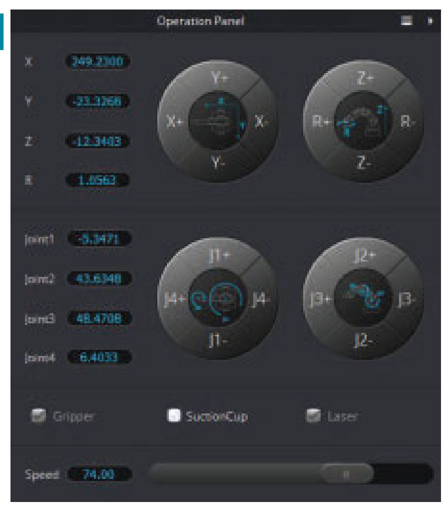


Figure 11.

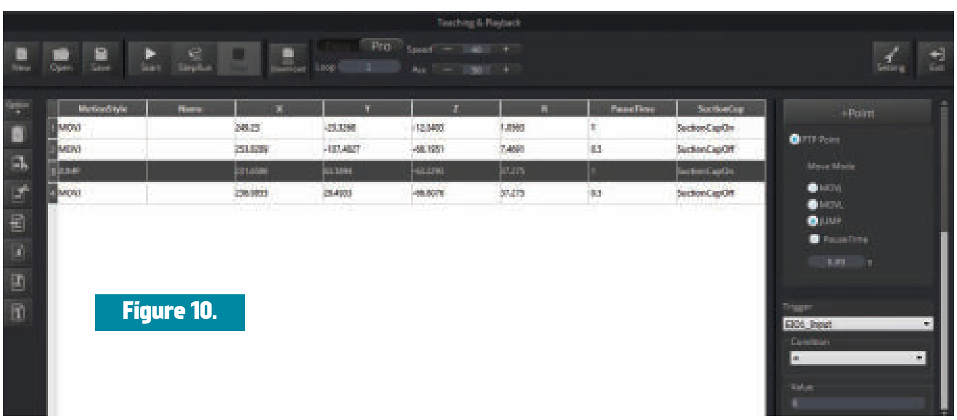


Figure 10.

System on a Chip
Simplifies Servo Control

A programmable system on a chip lets you quickly create PWMs as needed, and control them with only a few commands.

Control of digital servo motors – servos — can seem quite complicated. Thousands of microcontrollers include pulse-width modulator (PWM) circuits that will control servos. Unfortunately, the PWM timers, comparators, and registers that control them can baffle even expert programmers. I recently used a Programmable System on a Chip (PSoC) device from Cypress Semiconductor to overcome these problems. A PSoC IC contains only a few hardwired peripherals such as A-to-D (analog-to-digital) and D-to-A (digital-to-analog) converters, op-amps, and a real time clock. When designers and programmers need UARTs, and SPI and I^2C ports, they rely on design software that creates them from internal Universal Digital Blocks (UDBs). Think of these blocks as programmable logic circuits. If you no longer need a function, the UDBs become available for other operations.

The free PSoC Creator software tools simplify the design steps and help cut down development times. Each PSoC function provides a graphical user interface (GUI) that lets you quickly set most parameters without the need to write code. You also have direct access to a complete datasheet for each function and project examples.

Devices in the Cypress PSoC 5LP family include an ARM Cortex-M3 processor and Flash, read-write, and EEPROM storage for programs and data. Experimenters, hobbyists, and professionals can learn about PSoC Creator designs with the free design software and a $10 CY8CKIT-059 PSoC 5LP Prototyping Kit (**Figure 1**). You will need a short USB cable with a Type-A male connector on one end and a Type-A female receptacle on the other. Or, you can plug the kit right into a PC's USB port.

The CY8CKIT-059 board has its own information page that includes links to documents and software. The Cypress software runs under Microsoft Windows and on iOS computers that can emulate Windows PCs. (I used Windows 7.)

How does a PWM Controller Work?

At its simplest, a PWM comprises a down-counter, a comparator, a flip-flop, and a clock source as shown in **Figure 2**. (Later, you will learn how to use the PSoC Creator software to build a PWM project.)

This example uses a 16-bit binary (2^{16}) counter that starts at 65,535 and counts down to 0. This *start value* and the *clock frequency* determine the repeating *period* for the PWM output. You load the *compare register* with a value, say, 20,000. When the counter value decreases to 20,000, the comparator detects the equality and the flip-flop output changes state. When the counter reaches 0, the flip-flop resets, and the counter restarts again with a *start count* of 65,535. The compare register remains set at 20,000. The PWM function goes through this cycle again and again. In this case, the flip-flop would produce a logic 1 output for 20,000/65,536ths of the PWM period, or about 30.5 percent of the time (**Figure 3**).

In a PSoC PWM function, programmers

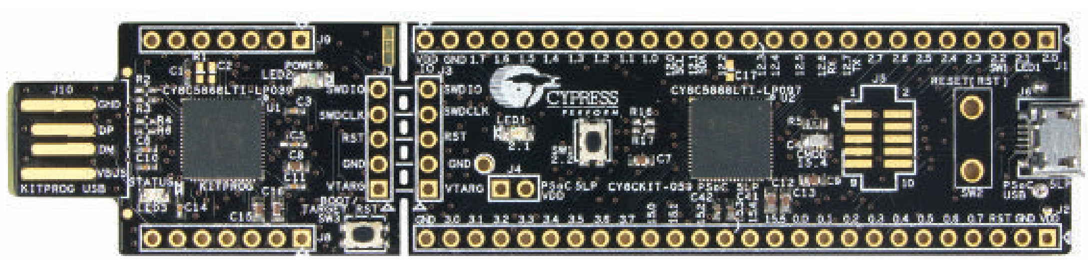


FIGURE 1. A CY8CKIT-059 PSoC 5LP Prototyping Kit. The 26 pins on each long side let you connect circuits to I/O ports, power, ground, and internal peripherals. To experiment with this module, solder pins to the contacts, cut a solderless breadboard down the middle, and plug in the board across the gap. You then have four breadboard contacts per module pin. The four contacts on the left side plug into a USB connector.

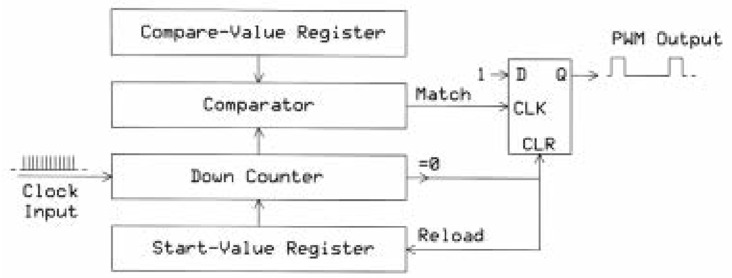


FIGURE 2. A conceptual PWM block diagram. The start value and the clock frequency determine the output period. The compare register holds a value to compare with the decreasing count. When the count and compare values are equal, the flip-flop changes state until the count reaches 0.

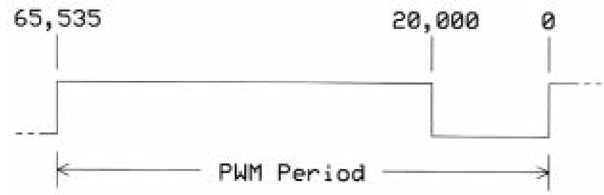


FIGURE 3. PWM output for a start count of 65,535 and a compare value of 20,000.

over a 180 degree swing. Resolution also depends on the type of load the servo must move. Servo manufacturer's datasheets don't always provide an angular resolution.

may set the upper starting count for any value within the 16-bit limit — 0 to 65,535 — so the counter doesn't need to start at its maximum value. To simplify the calculations that follow, I'll use a start-count value of 59,999 which requires 60,000 clock pulses to reach 0.

For servo control, I chose a 20 ms PWM output period based on datasheet information from servo manufacturers. (Values from 20 to 40 ms work with most digital servos, but always check period and pulse-width specifications before you use one.) During this 20 ms period, the PWM counter needs 60,000 pulses to count down from 59,999 down to 0. If the PWM requires 60,000 pulses in 20 ms, how many pulses will it need in one second?

$$\frac{60.0\ g10^{3}\ \text{pulses}}{20.0\ g10^{-3}\ \text{seconds}} = 3.00\ g10^{6}\ \text{pulses/second}$$

or a 3.00 MHz clock frequency

So, the PWM requires a 3.00 MHz clock signal. During the 20 ms period, a servo needs a logic 1 pulse with a width from 1.00 to 2.00 ms (**Figure 4**). The 1 ms logic 1 period drives a servo shaft to its counter-clockwise extreme and the 2 ms pulse drives it to its clockwise limit. A series of 1.5 ms pulses positions the shaft midway between the limits. As long as your controller continues to send pulses with the same timing, the servo shaft remains in one position.

Given the 3.00 MHz clock signal, you must calculate how many clock pulses will occur during a 1 ms logic 1 output period. That's the shortest logic 1 pulse a servo needs. Set up the ratio:

$$\frac{3.00\ g10^{6}\ \text{pulses}}{\text{second}} = \frac{x\ \text{pulses}}{1.0^{-3}\ \text{seconds}}$$

for 3000 pulses during a 1.0 ms period

If a 1 ms logic 1 pulse requires 3,000 clock pulses, then a 2.00 ms servo signal needs twice as many – 6,000 — loaded into the PWM compare register. In theory, you could have the servo shaft move to one of 3,000 possible positions, with values from 3,000, 3,001, 3,002 ... up to ... 5,998, 5,999 in the compare register. In practice, you might see an angular resolution of half a degree (0.5°)

Note: Depending on the servo in use, you might decrease the smaller compare value and increase the larger compare value to get a wider range of motion from the servo shaft. Consult your servo datasheets to determine limits and allowed pulse widths. Some servos allow continuous 360 degree shaft rotation.

Program a PSoC PWM

Right now you might think, "Wow, I have a lot of microcontroller registers to program, a clock frequency to create, and code to write!" Not so. The PSoC PWM function automatically uses UDBs to create a PWM that you configure via a GUI. Plus, the PSoC Creator produces a simple C language template to which you add only two operations: one to start the servo and another to set a servo position. It's that simple.

If you don't have a PSoC board, you can still create a PWM design with the Creator software. If you do have a CY8CKIT-059 PSoC board (or other PSoC-5LP board), you can program the chip and test a servo.

First, you need the CY8CKIT-059 Kit Setup Package that

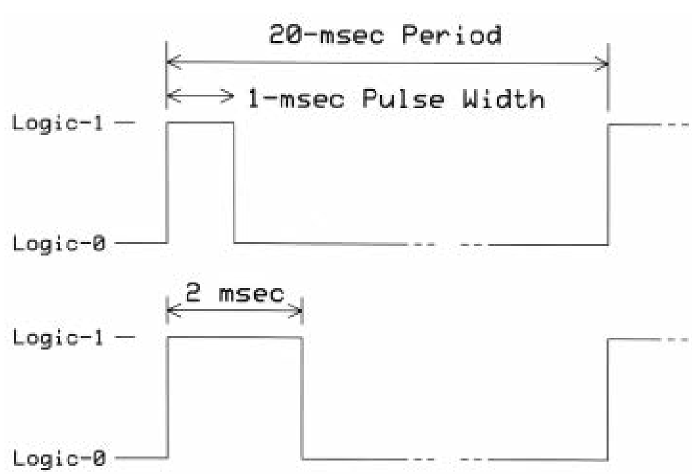


FIGURE 4. Logic 1 pulse widths for hobby-type servos generally vary from 1.00 to 2.00 ms within a 20 ms period. The output of logic 1 pulses will continue for as long as the PWM remains on.

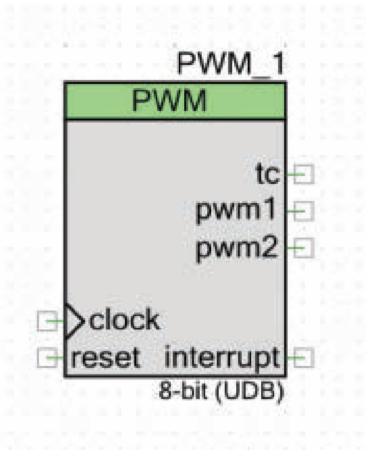


FIGURE 5. Schematic for a single PWM function, PMW_1. At this point, the PWM has gone through no configuration steps and has no components connected to it.

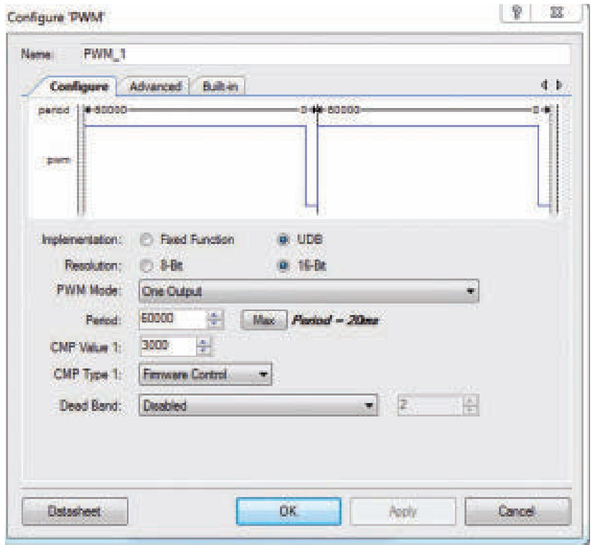


FIGURE 6. Parameters for the PWM_1 function in the schematic diagram shown in Figure 5. The parameter window also displays what the PWM output would look like on an oscilloscope.

includes the kit design files: PSoC Creator, PSoC Programmer, documentation, and examples. Download and install the software. When you first start PSoC Creator, click on Help and then on Update Manager to check for and handle updates you choose. You should update older versions of PSoC Creator to version 4.0 (or newer).

Note: When you start PSoC Creator 4.0, you might see a window, "Do you want to allow the following program to make changes to this computer," with the program name "cykeillic." Click "No." You do not use or need this program to utilize PSoC Creator for the 5LP devices.

I highly recommend you download the Cypress document, "Getting Started with PSoC 5LP," AN77759, which provides a good introduction to the PSoC architecture and a PWM example in Section 6. Follow the directions to learn how to start a project; you will create a PWM circuit that changes the intensity of an LED. If you have a PSoC 5LP board, load the program onto the board and watch the LED. (See the **sidebar**, "Caution: Firmware.")

To build a servo controller, start fresh with a new project. Again, follow the Section 6 instructions in the "Getting Started ..." guide. After you see a blank schematic window with only the layout grid, look on the right side of the PSoC Creator window and find the Component Catalog. Locate the PWM function under the Cypress—>Digital—>Functions heading. Click on the PWM label and you'll see a PWM block diagram. Drag the PWM symbol into the schematic area. The schematic will appear like the one in **Figure 5**.

Next, double-click on the PWM symbol on the schematic and a window will open with the PWM parameters (**Figure 6**). For now, ignore the timing diagram information because it will change.

In the Implementation area below the timing diagram, check UDB (for Universal Digital Block). The other settings are as follow:

Resolution: 16-bit
PWM Mode: One output
Period: 60000 (Key in this value and ignore any calculated period value)
CMP Value 1: 3000
CMP Type 1: Firmware control
Dead Band: Disabled

Click on OK. You have set up PWM_1 to control a servo.Go to the Component Catalog again and choose: Digital—>Logic—>Logic Low (0) and drag the "0" symbol into your schematic. Close the Digital list but stay in the Component Catalog. Go to System—>Clock and drag a clock into the schematic. Use the wire tool — a blue wire with a small square on each end — on the upper-left side of the schematic area to connect the "0" to the PWM Reset input, and likewise connect the Clock_1 block to the PWM clock input.

When you place the plus sign (+) wire-tool cursor over a connection point, the plus changes to an X, which indicates you can connect a wire there. If you make a mistake, click on the wire and delete it.

Double-click on the Clock_1 block and a Configure cy-clock window will open (**Figure 7**).

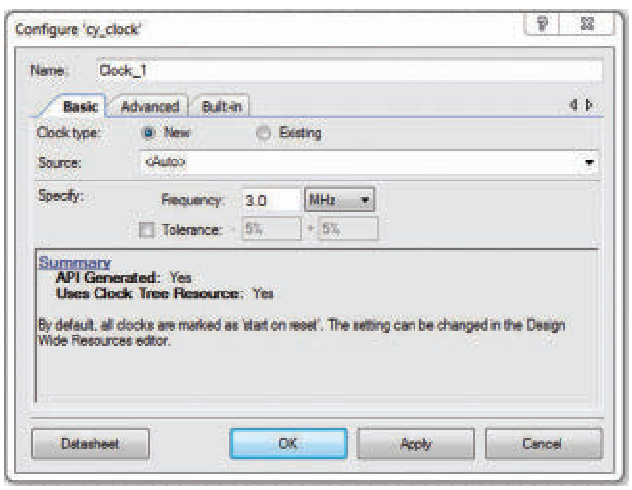


FIGURE 7. The configuration window used to set parameters for a clock signal.

Under the Basic tab, set:

Clock Type: New
Source: <Auto>
Specify Frequency: 3.0 and choose "MHz"

If you find a check in the Tolerance box, you may uncheck it.

Now click OK.

Go back to the PWM symbol and double-click on it to open its configuration window. The Period line should now show 60000 and a 20 ms period.

The timing diagram will show a short logic 0 pulse. If you do not see this information, recheck the PWM_1 and Clock_1 settings. Finally, click OK on any open configuration windows.

A short logic 0 pulse? But we need a logic 1 pulse! That's an easy "problem" for a PSoC chip to solve. We simply connect an *internal* inverter to the pwm output from the PWM_1 block. Within the Component Catalog, go to the Digital—>Logic section and click on the Not function. Drag the inverter symbol into the schematic and position it near the PWM_1 symbol's pwm pin.

Within the catalog, locate the Ports and Pins heading; open that list and drag a Digital Output Pin close to the inverter's output. This pin will connect to an external servo. (Question: If you could not use an inverter or other inverting device, how would you still generate positive pulses with the correct lengths? The answer is given at the end.)

Double-click on the Digital Output Pin to open its configuration window and give the pin the name

Caution: Firmware

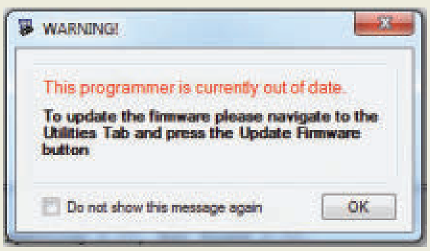


Figure SB-1. This warning notice alerts users to the need to update a PSoC board's firmware.

When you first connect a CY8CKIT-059 PSoC 5LP Prototyping Kit (or perhaps another PSoC board type) to your PC and create your first project, you might see a warning note that indicates the PSoC Creator: 1) Cannot connect to your board; 2) The programming did not occur or failed; or 3) That the board's firmware is out of date. If this happens, go to the list of programs on your PC and locate the "Cypress" folder.

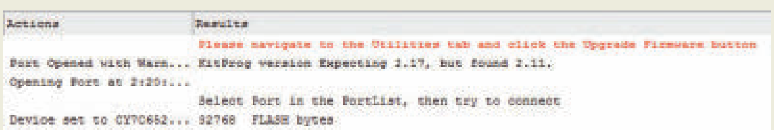


Figure SB-2. Instructions to update the firmware. "Please navigate to the Utilities tab and click on the Update Firmware button."

Inside you will see a folder labeled "PSoC Programmer" and in it the PSoC Programmer program. Run it. This program will let you update the board's firmware. **Figure SB-1** shows the type of warning message you might see. Don't panic. An update takes only a few minutes. Click on OK. The warning window will close. A message at the bottom of the window explains just what to do (**Figure SB-2**).

Figure SB-3 Location of the Utilities tab at the top of the Programmer window and the Update Firmware button underneath.

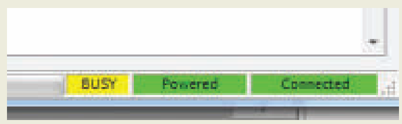


Figure SB-4. The yellow BUSY indicates the attached board is going through some operation.

After you click the Update Firmware button, you should see progress information in the bottom-right corner of the Programmer window (**Figure SB-4**).

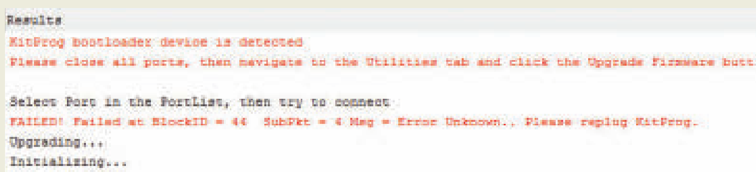


Figure SB-5. Oops, my first update attempt failed. What now? Another attempt!

When I first tried to do an update, the process failed (**Figure SB-5**).

So, I pressed the Update Firmware button again. Now the update started (**Figure SB-6**).

Figure SB-6. The increasing length of a green bar shows the update started and — so far — all is going well.

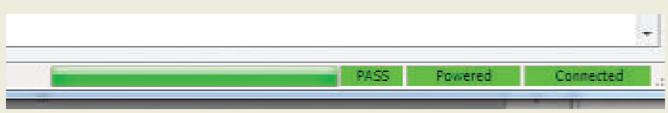


Figure SB-7. Success! The update finished, the board has power, and it has a good connection with the PSoC Creator software. Let's create a design!

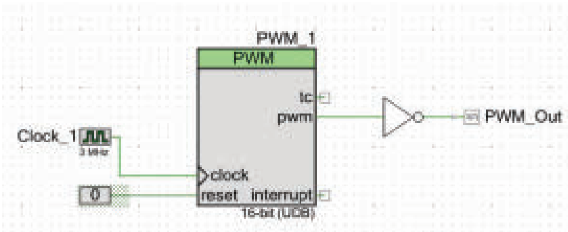


FIGURE 8. Final connections for the PSoC PWM servo-controller circuit.

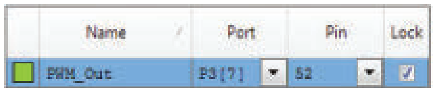


FIGURE 9. This view shows the single external pin created for the servo controller (PWM_Out). The pin assignment window lets you connect this signal to any available I/O pin. In this example, I used port 3, bit 7.

PWM_Out. Ensure the Digital-output and HW-connection boxes have check marks. No other boxes should have a check mark. If they do, click to uncheck them. Click on OK.

Use the wiring tool to connect the PWM_1 output to the inverter input and connect the inverter output to the digital I/O pin as shown in **Figure 8**.

The PSoC Creator needs to know which physical I/O pin on the PSoC chip to connect to the inverter's output. Cypress Semiconductor Application Note AN77759 explains the pin assignment process in detail, but here's a quick review. Within the left-side Workspace Explorer area, find the folder labeled "Design Wide Resources (*your_project_name*).cydwr" and double-click on it to get a top view of the PSoC chip on your PSoC board. In the upper-right of the screen, the PSoC Creator opens a pin assignment window where you will see a colored square and the names of I/O pins created in the schematic diagram (**Figure 9**).

In the Port column, click on the small down triangle to see a list of the chip's I/O ports. You can scroll through the list of all the pins. I assigned port 3, bit 7, labeled "P3[7] Opamp[3].vout."

This pin can connect to the output of a hardwired internal operational amplifier, but I selected it as a digital-output connection. In the Pin column, I saw the PWM signal routed to pin 53 on the IC package. You would assign other signals to physical pins in the same way.

You also may assign groups of pins. The P3[7] signal connects to a pin on the bottom edge of the PSoC board, toward the left. (The download file for this article includes a large pin-out chart, code, and two sample projects.)

Now, you can "build" the PWM application. Under the Build menu, click on "Generate application" and you will see messages scroll through the Output window (which you might need to enlarge) under the pin-out diagram. In the bottom-right corner of the screen, you will see a count of Errors, Warnings, and Notes.

You can view these in the Output window. You should have no errors.

After you build a PSoC application, click on the View tab at the top of the PSoC Creator window and then select the Resource Meter. The new view shows how much of the PSoC your design used. After I set up four PWMs, the meter showed 33 percent of the UDB were used.

The PSoC 5LP family of chips provides 24 UDBs, so you can create many PWM functions in one chip. That flexibility provides a lot of value. The PWM datasheet lists 31 application programming interface (API) software functions you can use in software to control a PWM block. As you'll see next, you can do a lot with only one.

Write the Code

To use the PWM_1 output, you must write *one line of code* to start the PWM operations. Under the Workspace Explorer heading on the left, find the main.c file about halfway down the list. Double-click on it to open a code template. Between the program line "/* *Place your initialization...*" and the *for(;;)* statement insert (refer to **Listing 1**):

```
PWM_1_Start();
```

That's it. You're done! Of course, looking at a 1 ms repetitive pulse on an oscilloscope screen isn't exciting, and the PWM_1 signal only moves your servo shaft to one position and keeps it there. Big deal. **Figures 10** and **11** illustrate the pulse-width value and the time between pulses on a Tektronix TDS-460A scope.

Listing 1.

```
/* ========================================
 * Jon Titus, January 8, 2017
 * Simple PSoC PWM output routine.
 * Pulse 1.5 msec pulse width set
 * in PWM configuration window.
 * ========================================
*/

#include "project.h"
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    // Initialization/startup code
    PWM_1_Start();

    for(;;)
    {
        // Endless loop
    }
}
```

PWM and Servos in Action

This short program increases and then decreases the PWM compare value in loops that cause a servo shaft to move back and forth. No need to rewrite the code from **Listing 1**; simply modify the main.c program you have.

Listing 2.

```
/* =========================================
 * Jon Titus, KZ1G
 * PSoC 5LP Servo demo
 * January 07, 2017
 * PWM clock set at 3.00 MHz
 * =========================================
*/
#include "project.h"

int main(void)
{
    uint16_t low_count = 3000;           // Count for 1-msec pulse
    uint16_t hi_count = 6000;            // Count for 2-msec pulse
    uint16_t pwm_count = 0;              // Counter for loops

    CyGlobalIntEnable;                   // Enable global interrupts.
                                         // Interrupts not used here.

    // Initialization code
    PWM_1_Start();                       //Start PWM_1 function

    for(;;)                              //Endless loop
    {
        pwm_count = low_count;           //Start compare value
                                         //for a with 1-msec pulse
        while (pwm_count < hi_count)     //Run loop until 2-msec pulse
                                         //time
        {
            PWM_1_WriteCompare(pwm_count);  //Output PWM compare value
            CyDelay(1);                  //Delay 1 msec for scope trace
            pwm_count = pwm_count +1;    //Increment compare value
        }                                //while loop ends

        while (pwm_count > low_count)    //2-msec count reached
        {                                //start to count down to 1-msec
                                         //compare value
            PWM_1_WriteCompare(pwm_count);  //Output PWM compare value
            CyDelay(1);                  //Delay 1 msec
            pwm_count = pwm_count -1;    //Decrement compare value
        }                                //while loop ends
    }                                    //for loop ends
}                                        //main.c ends
```

This code sets the starting count for the PWM, as well as the count for a 1 ms and 2 ms logic 1 pulse. The two loops increase the count to 6000 and then decrease it to 3000. Each new count gets loaded into the PWM compare-value register. Refer to **Listing 2**.

As explained earlier, after you start a PWM, you only need the *PWM_x_Write Compare (number);* instruction to move a servo to a specific position. No registers, interrupts, service routines, or bits to program, test, or reset.

Of course, the PWM API functions give you more capabilities, but use them only as you need to. **SV**

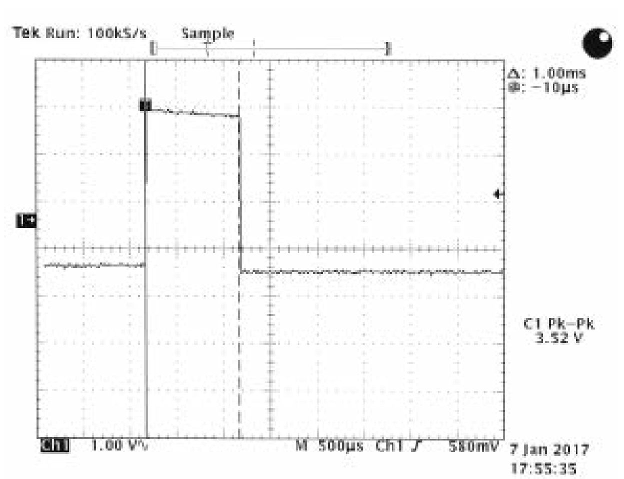


FIGURE 10. A 1 ms pulse produced at the PSoC PWM_1 output with a compare value of 3000. The vertical cursors let the scope measure the time between two points: the pulse edges.

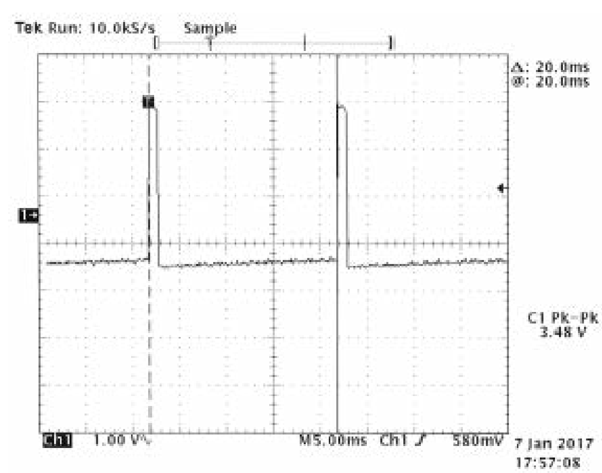


FIGURE 11. This scope image shows two 1 ms pulses spaced 20 ms apart as measured by two vertical cursor lines.

Answer: The PWM produces a logic 1 on the pwm when it starts to count down. Set the compare at 60,000 minus 3,000 or 57,000 for a 1 ms pulse, and to 60,000 minus 6,000 or 54,000, for a 2 ms pulse.

PSoC is a registered trademark of Cypress Semiconductor.

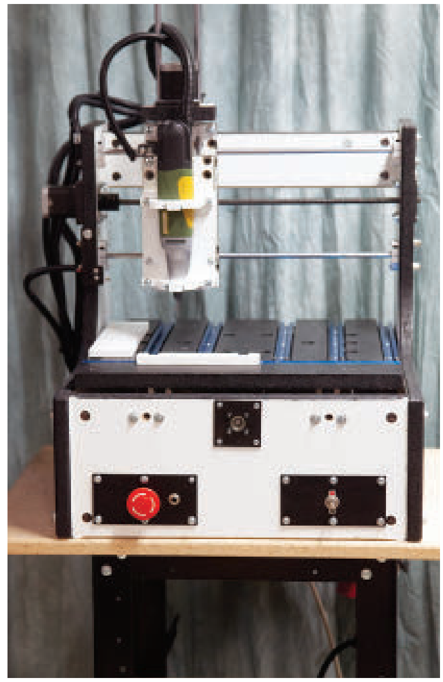
Resources

CY8CKIT-059 PSoC 5LP Prototyping Kit
http://preview.tinyurl.com/gl64yuf

PSoC software compatibilities with Windows and iOS
www.cypress.com/knowledge-base-article/apple-and-linux-os-support-psoc-software-and-kits-kba87545

Building the KReduCNC Conclusion

By Michael Simpson



In this final installment, I am going to finish the KReduCNC build, taking you through its operation by creating a part. Before we start, I want to go over some of the upgrades that I have added to my KReduCNC. Please note that I will be creating instructions for these upgrades on my website in the months to come.

Post comments on this section and find any associated files and/or downloads at **www.servomagazine.com/index.php/magazine/article/March2017_Build-CNC-Conclusion**.

Emergency Stop

The first upgrade I add to any CNC I build is an "Emergency Stop" or Estop as it is commonly referred to. The Estop shown in **Figure 1** allows me to quickly shut down the CNC operation if things go south. Some controllers have inputs designated for the Estop. These allow the hardware to shut down and then send a signal to the software (Mach3) to stop the job.

Other controllers have generic inputs that can be assigned in Mach3 as an Estop. This will enable you to quickly halt the job in software. Until you get an actual button wired, you can use the large Mach3 reset button located in the lower left of the application.

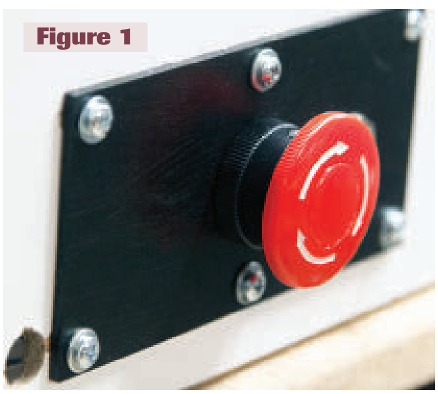


Figure 1

Figure 4

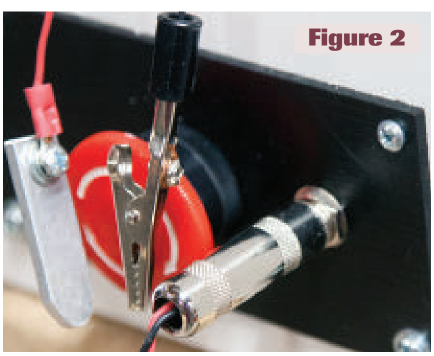


Figure 2

Figure 3

Depth Probe

One of the more tedious aspects of setting up a CNC to get it ready to mill your part is setting the tip of your bit to the surface of the stock. The depth probe shown in **Figure 2** allows you to set the bit by slipping a pad under it and hitting a button.

Power Switch

By adding a power switch like the one shown in **Figure 3**, it can be used to power down your CNC when it is not in use. This is a safety feature that will also extend the life of your electronics.

Homing Switches

If you add a set of homing

switches like the ones shown in **Figures 4**, **5**, and **6**, you can set up the minimum and maximum travel of your machine. This will help you keep from crashing it. This will also allow Mach3 to check any G-Code you load to make sure it does not machine outside the boundaries.

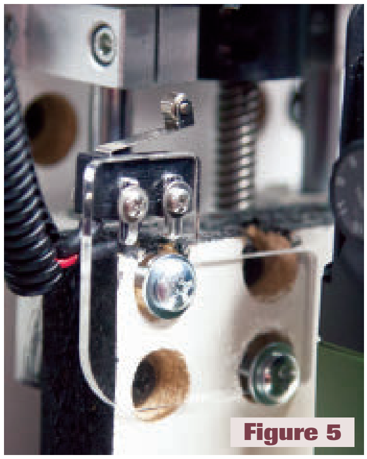


Figure 5

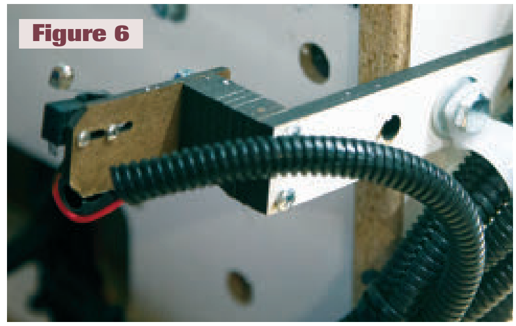


Figure 6

Fences

Once you have a set of homing switches installed, a set of fences like those shown in **Figure 7** will allow you to easily align your stock consistently and accurately. Once tuned, you can even flip your stock and machine the back side of it if needed.

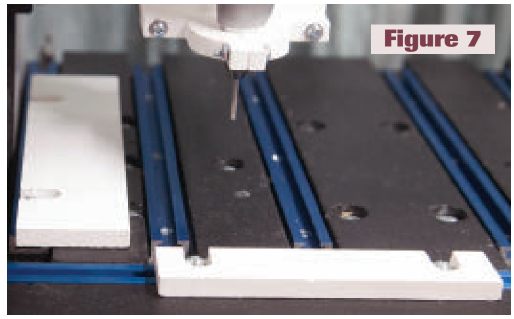


Figure 7

Motor Control

The controller I used in the KReduCNC has a relay built in. By wiring a set of connectors to this relay, they can be used to turn your spindle on and off automatically.

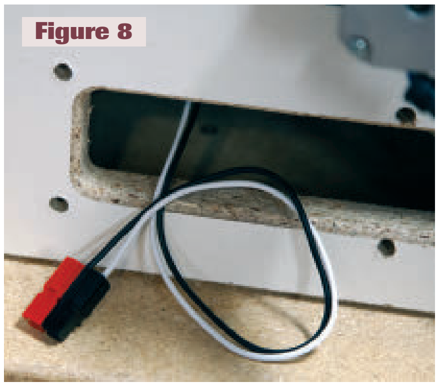


Figure 8

KReduCNC Operation

I want to take you through the operation of the KReduCNC by showing you how I made a set of special tool holders shown in **Figure 9**.

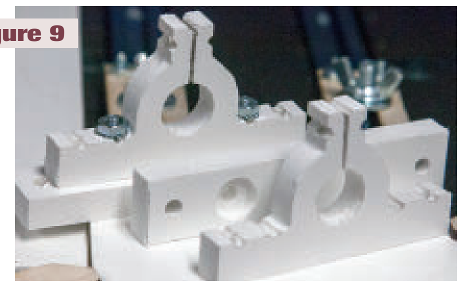


Figure 9

CAD

Many times when we make a part, we don't do it in full 3D. It is cut out of a piece of stock following a set of toolpaths that we create with CAM software. This is often referred to as 2.5D. When I design a part in 2.5D, I use a CAD package like CorelDraw (**Figure 10**) to lay out the part. Another CAD package that works well with 2.5D design is Adobe Illustrator.

CAM

Once the part is designed, it is exported into a format that the CAM software can understand. In this demonstration, I will be using Vcarve Pro by Vectric. I have found that exporting the part in EPS format works very well with this program.

The job of the CAM software is to create a toolpath for each graphic vector we want to cut or drill. The CAM process is started by importing the exported CAD file into

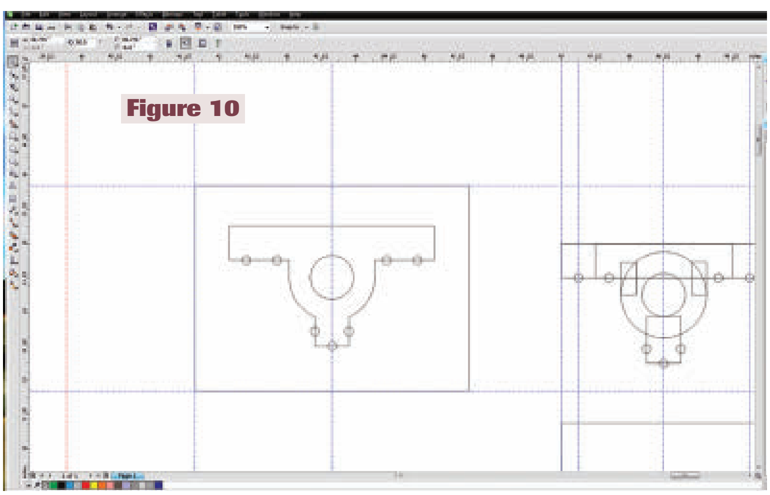


Figure 10

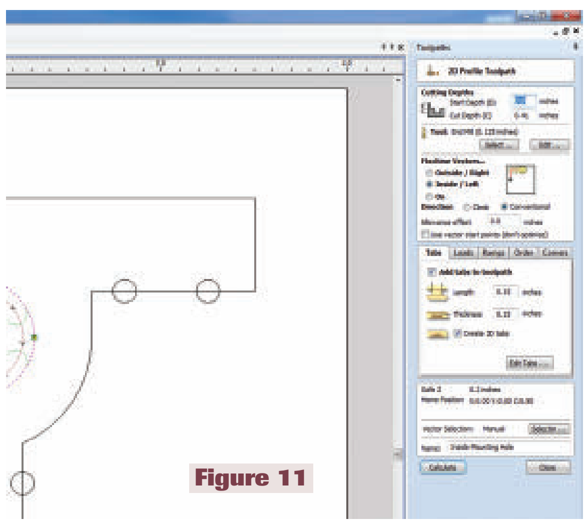


Figure 11

Vcarve. Vcarve has various options for creating a toolpath. The profile toolpath shown in **Figure 11** is the most common.

When creating a profile toolpath, you tell the software which graphic vector to use, how deep, which tool, tool speed, and whether it's an inside or outside cut. Another option when creating toolpaths in Vcarve is adding tabs. Tabs allow you to cut all the way through the stock, but still leave it attached.

Vcarve will overlay your toolpaths on the drawing (**Figure 12**) so that you can keep track of what you have done. In addition, as each toolpath is created, you can view the actual cut part in the simulation shown in **Figure 13**. Once all the parts have been created, the toolpaths are exported as G-Code. G-Code is what Mach3 uses to move the spindle in order to machine the part.

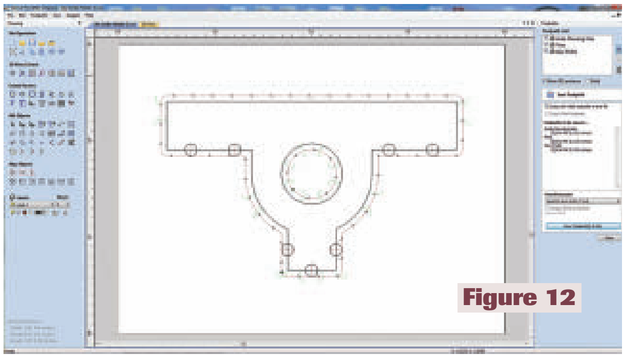


Figure 12

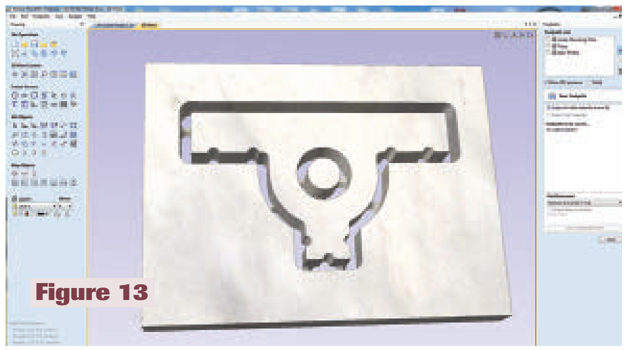


Figure 13

Stock Setup

The design calls for a piece of stock 4" wide by 3" tall and 3/8" thick. In addition to the stock, you will need a waster board cut to the same size as shown in **Figure 14**. The waster board is used to keep your bit from hitting your table top when the CNC cuts through the stock.

Clamp the stock and the waster board to the table as shown in **Figure 15**. You can use any piece of wood with a hole in it as a clamp. The T-slots are designed to hold the head of a 1/4" hex bolt. Just add a washer and 1/4" wing nut and any piece of plywood or solid wood with a hole in it, and you can secure the stock. Leave the clamps a little loose and the left edge free as you will be squaring the stock in the next step.

Figure 14

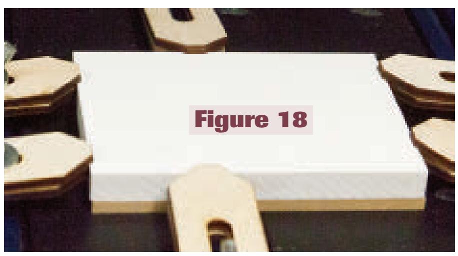


Figure 18

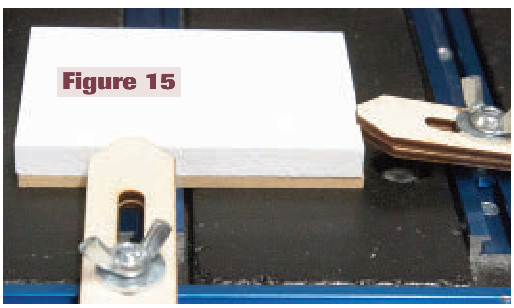


Figure 15

Figure 19

Let's square up the stock then. This is done by placing the edge of the bit up against the side edge of the stock as shown in **Figure 16**. The bit should be near the front of the stock. Next, move the spindle straight back to the rear of the stock as in **Figure 17**. You want the edge of the stock up against the bit. If it's not, move the stock until it is. Go back and check the front again by moving the spindle forward. If the stock is no longer against the bit, move the stock until it touches the bit.

Keep repeating this process until the stock is against the bit at both the front and back of the side edge. If you add the homing switch and fence upgrades, this step is pretty much eliminated as the edge of your stock will always be square against the fence.

Add the rest of your clamps as shown in **Figure 18**. Use as many clamps as it takes to secure the stock. The clamps need to keep the stock flat against the table, and keep it from moving from side to side. If you look closely, you will notice that the clamps are two pieces of ply. The ply on the bottom of the clamp is slightly shorter than the one at the top.

Figure 16

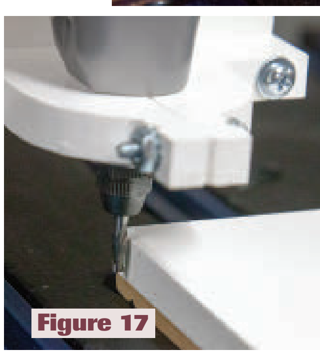


Figure 17

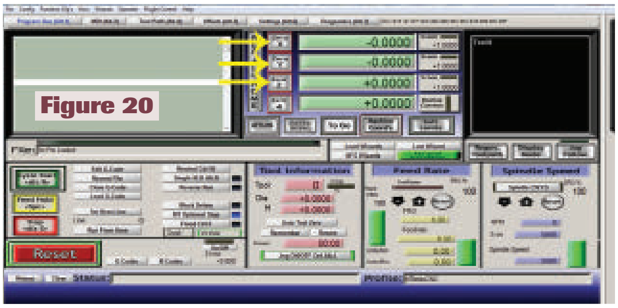


Figure 20

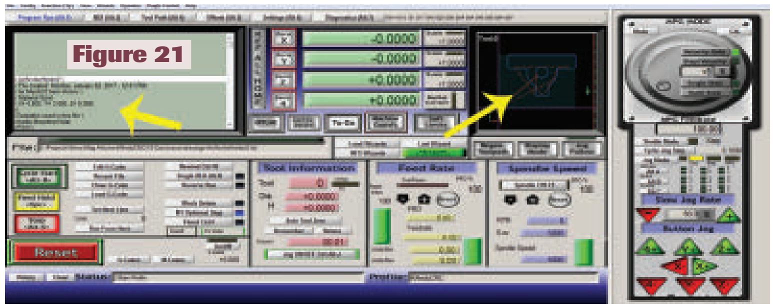


Figure 21

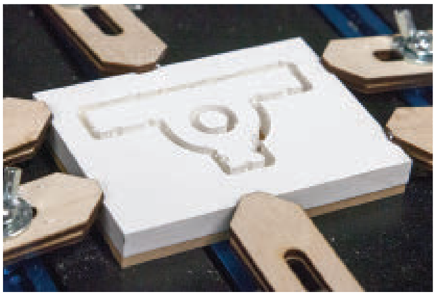


Figure 22

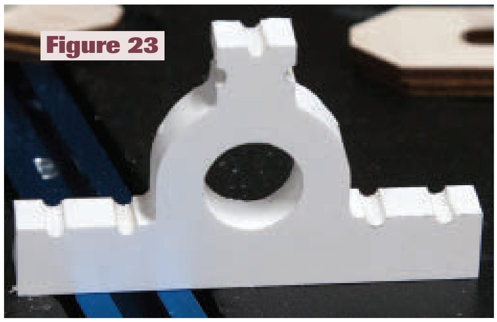


Figure 23

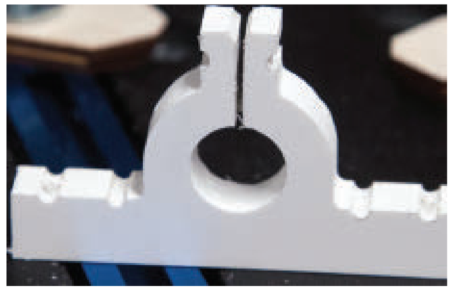


Figure 24

This creates a lip that keeps the stock from sliding. If you are using solid clamps, you can grind a small notch in the clamp to accomplish this.

One thing to bring up here while I am talking about clamps is that I always use wood clamps on my CNC routers. This is so I don't ruin the bit if I accidently hit one while machining the part. Believe me, you will hit the clamp from time to time.

To align the spindle with the stock, move the bit so that its center is directly over the front left corner as in **Figure 19**. Move the spindle down so that the bit is just touching the top surface of the stock. Once the bit is in position, go to Mach3 and zero the X, Y, and Z axes as shown in **Figure 20**.

Machining the Part

The first step in machining the part is to load the part's G-Code into Mach3. This is done with the File/Load G-Code menu option. Once the G-Code has been loaded, your Mach3 screen should look like the one shown in **Figure 21**. The actual code will be displayed in the panel on the upper right side of the screen. Mach3 also shows the actual toolpaths in the upper right panel.

You can find all the support pages for this build at **www.kronosrobotics.com/kreducnc**. You can see a video of the KReduCNC machining the tool holder at **https://youtu.be/S3WJdL5VAek**. There is also a short clip of the pneumatic engraver in action at **https://youtu.be/9aP2tdZBWEM**. As always, for any questions or comments, visit the *SERVO Magazine* forums at **http://forum.nutsvolts.com/viewtopic.php?f=49&t=17408**. Be sure to share both your good and bad experiences with this build, as both will help fellow readers.

To start the machining process, turn on your spindle and set its rotation speed. In this case, I set the spindle to about 18,000 RPM. Failure to turn the spindle on before starting the cutting operation could result in a broken bit. Note that the spindle control upgrade will allow Mach3 to turn on the spindle once the job starts.

To start the job (cutting process), hit the large green button labeled "Cycle Start."

Finishing the Part

Once started, Mach3 will direct the spindle to cut each of the toolpaths. The finished part will look like the one shown in **Figure 22**. Well, not exactly. Unless you add some sort of dust collection upgrade, the part will be covered in shavings and dust. The first step in finishing the part is to remove the stock and vacuum all the debris. This is very important as any debris left on the table will affect how the next piece of stock sets up on the table.

The part is still attached to the stock with the small tabs added with the CAM software. Punch the part out of the stock. You can remove the small tabs with various methods. Wire cutters will work, but I prefer a small rotary tool with a sanding or carving bit.

Depending on the material, you may need to remove flashing from the edges of the part. Most of the time, I use a fine sanding sponge. Once cleaned, the part will look like the one shown in **Figure 23**.

You may have noticed there are some small grooves in the part. These were added as guides to drill some mounting holes. One of the grooves is also used as a guide to cut a slit in the front of the part. The finished part is shown in **Figure 24**.


Figure 25

There are two of these parts used to hold the tool I will be testing. In addition, these parts were meant for another CNC, so I made a couple of adapters that let me use them on the KReduCNC. All these parts can be seen in **Figure 25**.

Just for Fun

I mounted the tool holders on the KReduCNC, then added the tool. The tool is a pneumatic air scribe (engraver) shown in **Figure 26**. The air scribe is used to mark metal and other materials without actual machining. Looking at **Figure 27**, you can see that it works very well.

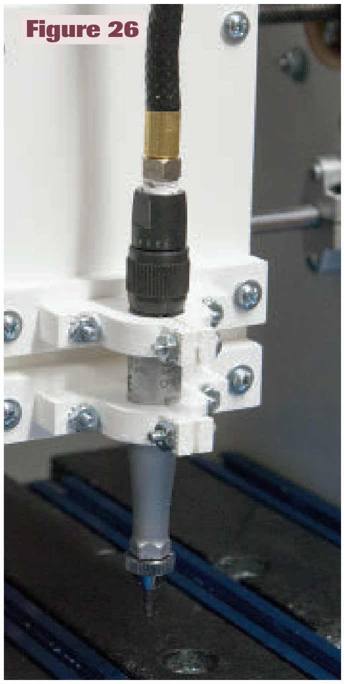


Figure 26

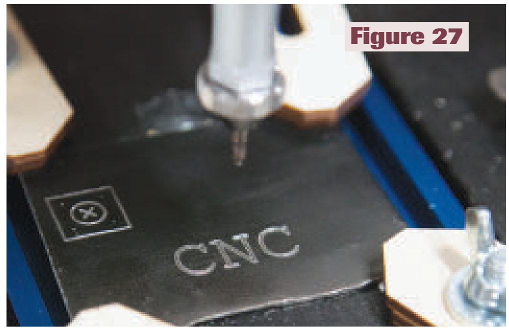


Figure 27

Conclusion

This completes the KReduCNC build series. I will post the drawing files for the tool holder on my support page for the KReduCNC.

As mentioned, I will be adding various upgrades to this build as time permits. **SV**

 Continued from page 21

HEY MYKIE — WHAT'S FOR DINNER?

With connectivity being a major theme these days, kitchen appliances are no exception. At the 2017 CES, Bosch announced both a connected fridge and a connected oven. While an oven that you can control remotely and a fridge that knows what's inside it are handy enough by themselves, the whole point of a kitchen is to use the food you have in combination with the necessary appliances to create tasty and nutritious meals. Mykie (short for "My Kitchen Elf") is a little countertop robot that Bosch developed in order to help tie your kitchen hardware together to recipes to make cooking easy and fun.

Mykie is a product concept right now, designed to be a personal kitchen assistant that can interface with the other appliances in your home. It's intended to be an embodiment of a smart kitchen — rather than talking directly to your stove or your fridge (which might be weird), you can talk to Mykie who will listen to you, talk back, and then control the rest of your connected appliances for you.

In addition to voice control, Mykie has a touchscreen you can poke at, but its most useful feature has to be a powerful little projector in his butt. Or, where his butt would be if he had one. Rather than having to rely on Mykie's little screen, you can just set the robot on the counter, and it'll project a much larger image onto your kitchen wall.

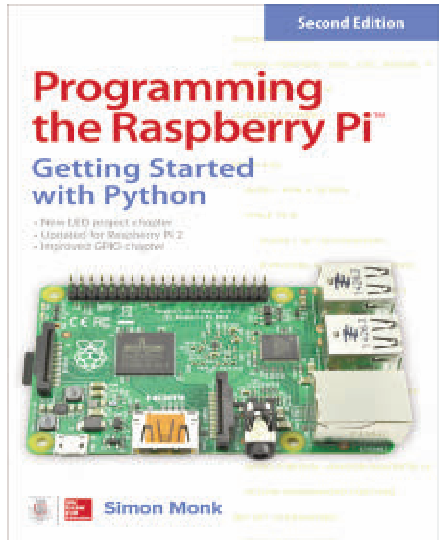
Since Mykie is a currently a product concept, Bosch isn't sharing any information on price or availability. It seems reasonable to expect that they're not just designing and building robots for fun, though, so Mykie just might show up in kitchens within a year or so (hopefully).

Photo courtesy of Evan Ackerman /IEEE Spectrum.

The *SERVO* Webstore

CD-ROM SPECIALS

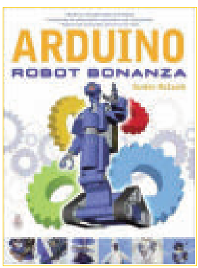
03.2017 FEATURED



ROBOTICS

Arduino Robot Bonanza

by Gordon McComb



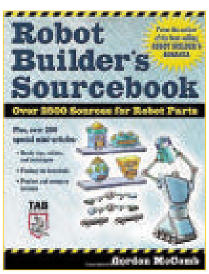
Create high-tech walking, talking, and thinking robots. "McComb hasn't missed a beat. It's an absolute winner!"

Breathe life into the robots of your dreams — without advanced electronics or programming skills. *Arduino Robot Bonanza* shows you how to build autonomous robots using ordinary tools and common parts. Learn how to wire things up, program your robot's brain, and add your own unique flair.

$29.95

Robot Builder's Sourcebook

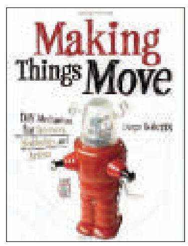
Fascinated by the world of robotics but don't know how to tap into the incredible amount of information available on the subject? Want the names, addresses, phone numbers, and web sites of companies that can supply the exact part, plan, kit, building material, programming language, operating system & computer system? Robot Buider's Sourcebook is just what you've been looking for!



$36.00

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts



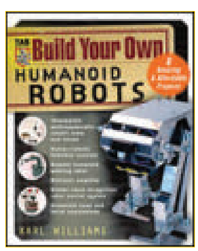
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95

Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!



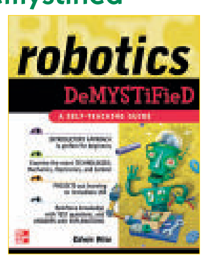
This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot. **$24.95**

Build Your Own Teams of Robots with LEGO® MINDSTORMS® NXT and Bluetooth®

Plan, design, assemble, and program robot squads that communicate and cooperate with each other to accomplish together what they can't do individually. Inside the authors show you how to construct a team capability matrix (TCM) and use the Bluetooth Robotic-Oriented Network (BRON) so your robot teams can share sensors, actuators, end effectors, motor power, and programs. **$30**

Robotics Demystified

by Edwin Wise



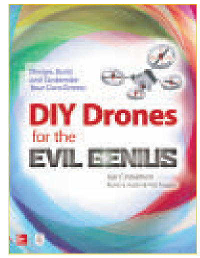
YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS! Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

To Order Call 800 783-4624 or visit our webstore

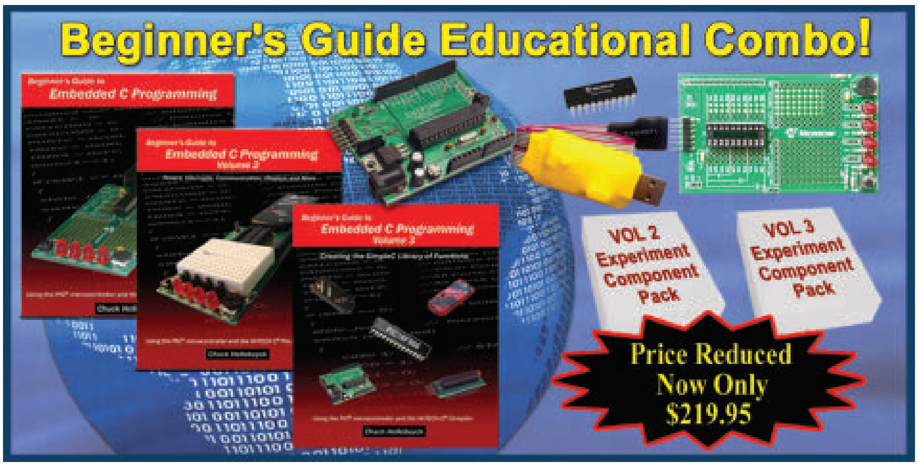
Order online
www.servomagazine.com

SPECIAL OFFERS

DIY Drones for the Evil Genius
by Ian Cinnamon, Romi Kadri, Fitz Tepper

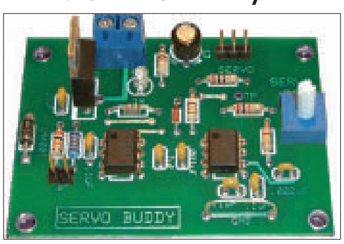


This fun guide shows step-by-step how to construct powerful drones from inexpensive parts, add personalized features, and become a full-fledged pilot. It not only covers safety, mechanics, drone design, and assembly, but also teaches the basics of Aerospace Engineering. You will discover how to add video transmitters, GPS, first-person view, and virtual reality goggles to your creations.
Reg $25.00 Sale Price $19.95

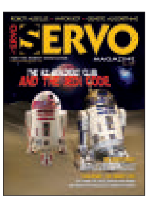


PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.



For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
$39.55

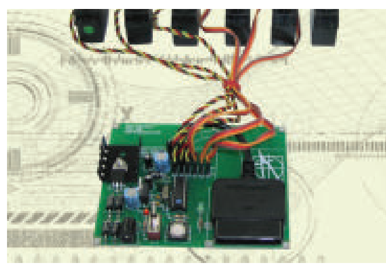
3D LED Cube Kit

From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue.
Jig and plastic cases also available.

$57.95

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.
For more information, please see the February 2011 edition of *SERVO Magazine*.
Assembled units available!

$79.95

The Multi-Rotor Hobbyist

Hacking the Cheerson CX-10

By John Leeman

Post comments on this section and find any associated files and/or downloads at **www.servomagazine.com/index.php/magazine/article/March2017_MultiRotorHobbyist_Cheerson-CX10-Hack.**

If there are children in your family or maybe just children at heart, it's likely that some of those small silver dollar sized quadcopters were buzzing around the house over the holidays (**Figure 1**). When first learning to fly these little copters, there are a lot of crashes. Fortunately, the quads are surprisingly resilient. Eventually, we end up with a few still working and a few "spare" parts laying around. Looking at my pile of old electronics accessories, I got an idea. Could I take an old computer joystick (**Figure 2**) and integrate it into the remote control of the Cheerson CX10? It would be like a flight simulator, but with something real flying around the room!

Figure 1: The CX-10 is the perfect size to fly around indoors, and made a popular holiday gift for quad enthusiasts.

Figure 2: An old Microsoft SideWinder joystick turned out to be the perfect controller for this project.

Introduction

It really is amazing that for around $15, you can have a complete four-channel quad delivered to your door in a couple of days (**http://amzn.to/2g6Ydpj**). For a little more money, there are even versions with cameras, more rotors, or other features. For my hardware hacking, I decided to stick with the basic model (**Figure 3**) as I already had a few that my wife and I used to "drone race" around the house.

During one of our flying battles through the house, one of the quads hit a wall. This event was not unusual, but this time it didn't survive. One of the four motors would no longer run. I had been looking for an excuse to open these up and come up with some fun modifications — this was my chance! I have always enjoyed flight simulator games and decided that attaching my flight simulator joystick to the Cheerson would be like the game, but better!

Like many projects, this one languished on a corner of the bench for a while. I needed to sit down and really figure out how to get into these things without destroying them, determine how the remote worked, and see if I could fix the damaged drone. Before I got around to it, Elecia White from the **Embedded.fm** podcast had taken the quad and remote apart, and done a series of really nice write-ups on it (**http://embedded.fm/blog/takingaparttoys**).

Looking at her articles gave me the boost I needed to drag the parts out of their back-of-the-bench exile and start figuring out how I was going to accomplish this project. Elecia also pointed me towards the work Alvaro Prieto had been doing on reverse engineering the ProtoX radio protocol used by the controllers (**https://github.com/alvarop/protox**). With several ideas germinating, it was time to get to it, starting by fixing the broken copter.

Quad Construction and Repair

Taking a look at the outside of the quad, it is pretty easy to tell it is built down to a price. No extra plastic anywhere, no extra fasteners; just the basics. The body consists of two plastic shells that make up the upper and lower halves of the body. To get started, I removed the propellers. They are press-fit onto the motor shaft, so ideally, you would just grab them and remove them.

My propellers were a little more stubborn. Using two small common blade screwdrivers, I was able to pry them off the motor shaft.

After that, I flipped the quad upside down and removed four small Phillips screws that were holding the shell together (**Figure 4**). The last thing holding the case together are some molded plastic clips at the ends of the arms by the motors. A few of my CX10s don't have all of theirs anymore!

Prying them off was easy with just my fingers, but you could go back to the screwdriver if you'd prefer.

Once opened up, you'll see that the airframe is actually the PCB (printed circuit board) of the quad. There are holes for the motors, and all of the sensors and processor take up less than a square inch. I'm not going to go through the board layout in detail as I didn't modify it, but do check out Elecia's articles for an in-depth analysis.

As soon as I got the case off of the quad, the cause of the motor failure became obvious. One of the wires running to the problem motor had been sheared during the impact with the wall. Looking at

Figure 3: The CX-10 has a propeller center spacing less than the length of an AA battery. After-market propellers are available in many colors to make it easier to distinguish the orientation of the quad, and differentiate between multiple quads in flight.


Figure 4: Removing four small Phillips screws is all it takes to pull apart the halves of the copter.

how the cables were routed, it did seem that the plastic case posed a pretty significant pinching hazard to the wires. With a magnifier, a little bit of patience, and some electrical tape, I was able to split the wire back together and try to reroute it slightly. I reassembled the quad and powered it up. It flew like new and we all know what that means: It's time to hack it apart.

Remote Construction

The remote (**Figure 5**) is a two-piece relatively simple design. To take the remote apart, remove the two Phillips screws at the back of the housing (**Figure 6**). The controller separates easily with the battery holder staying integrated with the back half of the enclosure.

Be careful of the wires running from the battery compartment to the circuit board. I went ahead and snipped them as I knew I'd be powering my setup from a different power supply anyway.

Once into the controller, the design is pretty straightforward. The joysticks of the controller are mechanically linked to four potentiometers that detect the position of each control (**Figure 7**). The left stick is spring loaded for the yaw axis, but the throttle stays wherever it is positioned. The right stick is spring loaded in both axes. Basically, it is set up just like the normal flight remote we are used to.

The potentiometers are set up as voltage dividers. One end of the potentiometer is connected to the battery voltage of ~3 VDC; the other is grounded. The moving wiper serves as the tap or connection between the two resistances. If

Figure 5: The remote is very simple, with two joysticks, trim controls, and a power switch. Each joystick is also a button, with the left changing the controller sensitivity and the right enabling the flip trick.

Figure 6: Remove the two screws in the back of the controller to separate the halves. Some controllers seem to have a third screw holding the battery door on, but it is not necessary to remove this screw to take apart the controller.

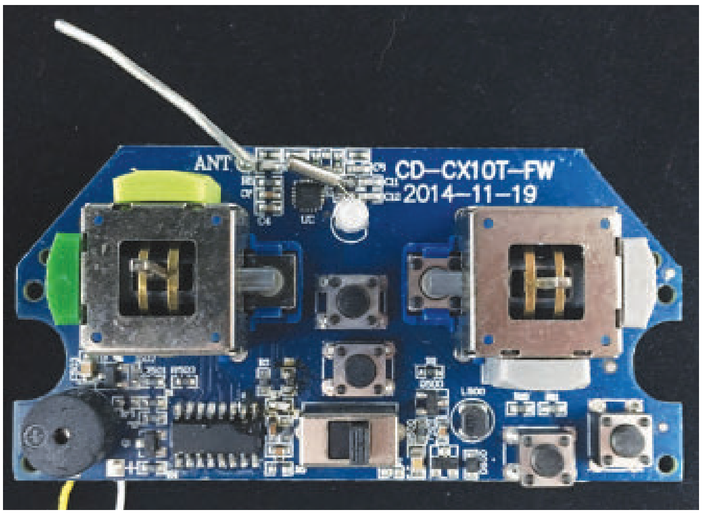


Figure 7: The construction of the controller is relatively simple. Two mechanical joystick assemblies manipulate four potentiometers. The trim and joystick buttons are clearly visible as well.

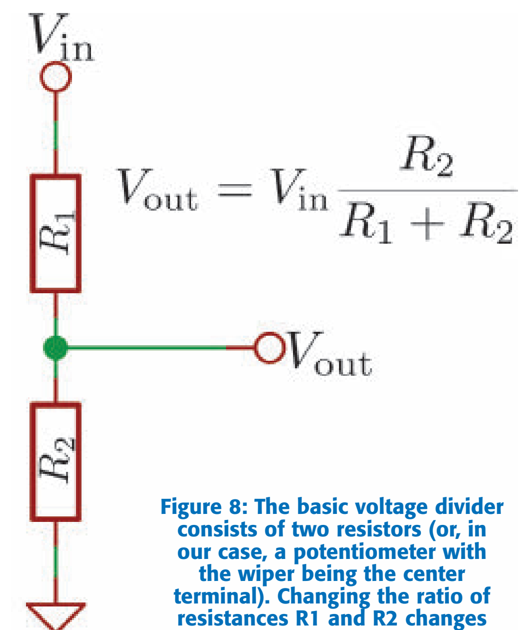


Figure 8: The basic voltage divider consists of two resistors (or, in our case, a potentiometer with the wiper being the center terminal). Changing the ratio of resistances R1 and R2 changes the fraction of the input voltage available at the output terminal.

you look at the traditional voltage divider and its equation (**Figure 8**), and also a potentiometer set up as a voltage divider, the idea is pretty straightforward. Moving the sticks changes the resistances in the divider circuit, and therefore the output voltage changes. All of the movement potentiometers are linear over their range, but the throttle potentiometer is nonlinear in the center to make the quad easier to control.

The remote has buttons to trim the controls, and there are buttons to detect presses on each of the control sticks (**Figure 7**). The left button changes the sensitivity of the controls, and the right button enables a trick mode allowing you to perform flips with the quad. After a little bit of investigation, I found out that the buttons pull the input pins of the microcontroller low — a useful bit of information to know later.

Hacking the Remote

I was thinking about what I wanted to be able to control from the computer. We could wire up all of the remote's buttons, but, generally, I didn't touch the trim buttons that much. I decided that replacing the joysticks would be the primary goal, but adding the functionality of changing the sensitivity and doing tricks would be a nice bonus.

The first step would be to get the existing controls off of the board and then patch in my own. While you could cut traces, the easier path seemed to be desoldering all of the parts I wanted to replace and soldering in my own wires to run to a microcontroller. In the end, I'm not so sure that was true, but it did end up working out well.

Removing the joysticks turned out to be a bit harder than I anticipated. Each potentiometer had three connections, and the housing of the joystick itself was soldered to the board in four places (**Figure 9**). Getting all 10 solder joints molten at the same time was next to impossible. I removed as much of the solder as I could with suction and solder braid, but, in the end, I still

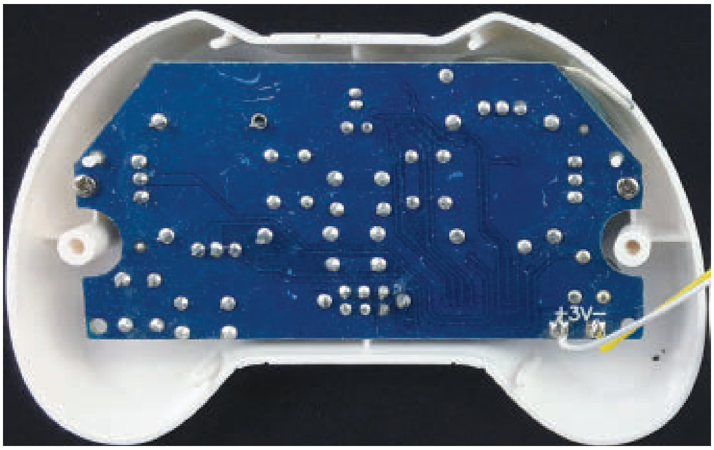


Figure 9: There are 10 solder joints that hold the joystick assemblies into position on the PCB: six for the potentiometers and four for the mechanical assembly. The soldering quality on these is questionable. Notice the completely dry joint in the upper left corner of the PCB.

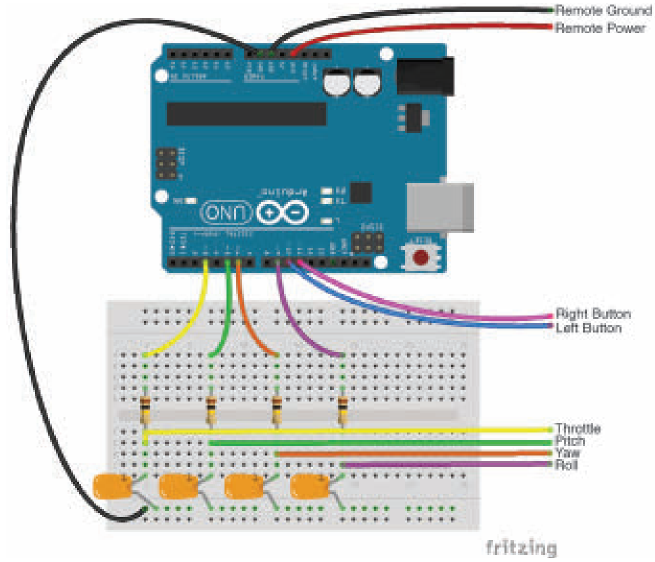


Figure 10: The hookup for this project is relatively straightforward. The roll, pitch, yaw, and throttle outputs are connected to the center terminal of the potentiometer footprint for each axis, respectively.

destroyed the joystick and potentiometers in the process.

Not a problem since I wouldn't be using them, but it felt like an excessively brute-force solution. Using a desoldering aid like Chip Quik® or having a proper desoldering iron would have made the job a little easier. After the fact, I was reading a blog post that suggested wrapping all the connections in heavy gauge copper wire and heating all of the joints at the same time.

Once the joysticks were out, I removed the two pushbuttons that they activated with relative ease, and began to look around at where all of the traces went. Careful inspection of the PCB will tell you a lot. You can determine if the voltage will be +3V or 0V when each stick is at its extreme limit. You can also determine what the buttons do when pressed.

This knowledge can also be gained by experimenting, but I like seeing how others design their PCBs. Sometimes you can pick up a good trick or two after looking at someone else's board layout.

Arduino Hookup

I decided to use an Arduino Uno for this project because most people have them in their box of development boards and the code is relatively easy to read. I implemented this solution with a Propeller microcontroller as well, but decided it was overkill in the end since we only need to do a few things with our microcontroller.

We need to output four analog voltages to simulate the positions of each control and we need to have two digital outputs to simulate the button presses.

The Arduino IDE (integrated development environment) does have a convenient *analogWrite* function built in. One would think that this means we can specify a voltage and the Arduino will output that voltage on the desired pin, but that isn't strictly true.

To do that in the strictest sense would require a digital-to-analog converter (DAC), but one is not available on the Uno. What really happens is that the output is switched on (5 VDC) and off (0 VDC) with a varying duty cycle. If we want 5 VDC out, the pin just turns on and stays on; for 0 VDC, it turns off and stays off. So, what if we want 2.5 VDC out?

We switch the pin on and off with a 50% duty cycle, meaning it is on half of the time and off half of the time.

The time averaged output voltage is then 2.5 VDC. That on-off time ratio can change up or down to make the average voltage over time at the pin equivalent to our desired output.

This switched square wave is running at around 500 Hz. If we connect an LED to the output and sweep through the range, we see a gradually varying brightness. That's fine if your system is not sensitive to those very fast switching cycles, but it's not good enough for our remote. I encourage you to try it and see the effect — it makes for challenging flying!

We need to filter out the high frequency switching and get closer to a true stable analog output voltage. This is a job for the humble RC filter. Yes, we could use an active filter, a multi-pole filter, or any number of solutions that have sharper response and would provide a lower impedance source for the remote, but remember, this is an afternoon hack. The equation for the corner frequency of a simple single-pole RC low pass filter is $F_c = 1/2\pi RC$.

I experimented with a few different filter combinations, but settled on using 100K Ω and 1 µF, which give us a corner frequency of just less than 2 Hz. Of course, there is a decent amount of higher frequency content making it through, as this corner frequency is the point of -3 dB attenuation.

Hook up the Arduino to the center pad on each potentiometer according to **Figure 10**. You can see my setup in **Figure 11**. I went ahead and hooked up the two buttons as well, since I knew I wasn't going to be able to resist implementing them.

Remember how I said this was an afternoon hack? It turned out that I didn't have any through-hole 1 µF capacitors left, but I did have some surface-mount parts. I soldered some bits of solid 22 GA hookup wire to the ends of the parts, and we were back in business. This is a trick I end up using a lot as my SMT parts catalog has begun to grow larger than my through-hole bin.

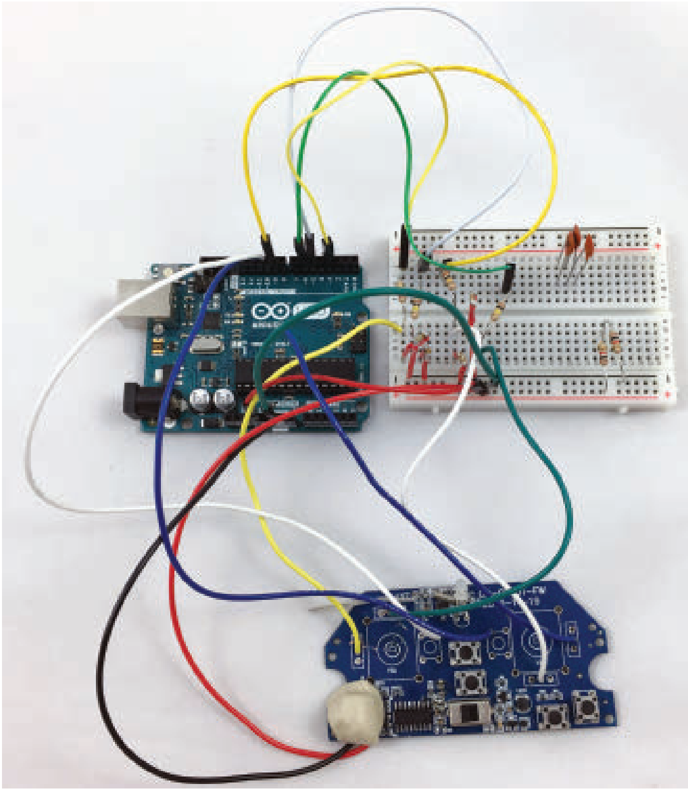


Figure 11: My quick prototype was not necessarily pretty, but it worked very well. Notice the putty over the buzzer to reduce its volume.

I also decided to block up the little piezo speaker with plumber's putty. I didn't want to remove it from the board since it is the only indication that we have successfully armed the quad, but it was just far too loud outside of the case.

Joystick Software

We need to be able to read a joystick and translate the control positions into commands that will be sent to the Arduino for output to the remote control inputs. My general tool for these type of input-output operations is Python. When you first install Python "out of the box" (who remembers buying boxed software?), there are many useful functions, but nothing for dealing with game controllers.

Luckily, the PyGame module makes our job easy! We'll quickly go through the setup of Python using Anaconda, install PyGame, and get a simple program up and running.

It is likely that your computer has some form of Python on it already. I prefer to use Anaconda distribution from Continuum Analytics in Austin, TX; it installs many useful packages and makes package and environment management much easier that doing it all yourself. Head over to **https://www.continuum.io/downloads** and download the appropriate installer for your operating system. Go through the install process as you normally would.

If you are going to run this project in a more storage constrained environment such as on a Beagle Bone or Raspberry Pi, consider installing the Miniconda distribution from the same page. It doesn't have all of the packages included, but you can install just those you need and save a lot of disk space.

On a regular machine, I would install the regular Anaconda distribution so you're all set for future experimenting and programming.

Next, we need to install PyGame. You will need to open up a terminal Window — use the Terminal app on a Mac or Linux machine. On Windows, I recommend using the Git BASH terminal that can be downloaded at **https://git-scm.com /downloads** (there is a BASH shell beta in Windows 10, but I haven't tested this procedure with it). Once at a terminal prompt, type *pip install pygame* and press return. PyGame will be installed automatically from here.

Next, install pySerial in a similar way. The serial library can be installed through the Conda package manager by typing *conda install pyserial* and another install will execute. If you are asked any questions about upgrading or downgrading packages, answer yes.

You can download the source code for this project from the article link or clone all of the code for this project from my GitHub repository at **https://github.com/ jrleeman/CX10_Controls**. There will be a few minor edits you will have to make to the Python source code later, so open up the *CX10_joystick.py* file in your favorite editor (try Atom from GitHub or PyCharm from JetBrains if you don't have a favorite yet).

At the top of the file, we import the libraries we are going to be using: PyGame and pySerial. We then initialize PyGame and open the serial port. When we program the Arduino later, we'll need to come back and update this serial port name. Don't forget to do this step, or you'll get errors. The serial port name for your Arduino can be found by looking in the *Tools -> Port* menu of the Arduino IDE.

Next, we enter a *while* loop that checks if we have

```
My Game

Number of joysticks: 1
 Joystick 0
  Joystick name: SideWinder Force Feedback 2 Joystick
  Number of axes: 4
   Axis 0 value:  0.056
   Axis 1 value:  0.048
   Axis 2 value:  0.016
   Axis 3 value:  1.000
  Number of buttons: 8
   Button  0 value: 0
   Button  1 value: 0
   Button  2 value: 0
   Button  3 value: 0
   Button  4 value: 0
   Button  5 value: 0
   Button  6 value: 0
   Button  7 value: 0
  Number of hats: 1
   Hat 0 value: (0, 0)
```

Figure 12: The SideWinder joystick I used has a lot of extra buttons and even a multidirectional "hat" control. These could all be adapted to control other features on a more complicated quad. The hat control seems like a perfect way to steer a camera gimbal.

created a quit event, then loops through the axes of the joystick and retrieves the position of each. This is another point of possible editing depending on your joystick, but we'll come back to that. We then read the status of two buttons on the joystick and finally write all of this information out on the serial port as a comma delimited string.

To determine if the joystick axis numbers are correct for your joystick and to see what buttons and features your joystick has, run the joystick demo app (obtained from the PyGame documentation at **www.pygame.org/docs/ref/joystick.html**) by typing *python joystick_demo.py*. This will create a pop-up window showing the output from every feature of your joystick that PyGame can access (**Figure 12**). If you notice that your throttle, pitch, roll, or yaw have different axis numbers from my joystick, be sure to change that in the *CX10_joystick.py* file.

Arduino Firmware

The Arduino sketch that we will use to read the joystick outputs from our Python program and turn them into PWM signals that we will feed into the remote control is relatively straightforward. I will briefly go over the basics though, so you can modify it as you see fit for your application.

At the top of the sketch, we define pin numbers for the throttle, pitch, yaw, and roll buttons. The analog controls need to use pins capable of generating a PWM signal to spoof the potentiometers we took off the controller. The buttons can use any digital pin as they just need to be able to switch on and off.

Before we go any further, it is critical to remember that this controller operates off of 3 VDC nominally. That means when the controls are at their most extreme limits, we need to output 3 VDC or so. Since we are powering the controller from the 3.3 VDC output on the Arduino, I'll

```
void setup() {
  Serial.begin(115200);
  pinMode(THROTTLE_PIN, OUTPUT);
  pinMode(PITCH_PIN, OUTPUT);
  pinMode(YAW_PIN, OUTPUT);
  pinMode(ROLL_PIN, OUTPUT);
  pinMode(LBUTTON_PIN, INPUT);
  pinMode(RBUTTON_PIN, INPUT);
  digitalWrite(LBUTTON_PIN, LOW);
  digitalWrite(RBUTTON_PIN, LOW);
}
```

Figure 13: In the setup function, we set the direction of each pin (input or output) and set the button pins to be low, even though we have them set up as high impedance inputs most of the time.

```
// Look through the serial string and pull out our values
throttleValue = map(Serial.parseInt(), MIN_IN, MAX_IN, MIN_OUT, MAX_OUT);
pitchValue = map(Serial.parseInt(), MIN_IN, MAX_IN, MAX_OUT, MIN_OUT);
rollValue = map(Serial.parseInt(), MIN_IN, MAX_IN, MAX_OUT, MIN_OUT);
yawValue = map(Serial.parseInt(), MIN_IN, MAX_IN, MIN_OUT, MAX_OUT);
leftbuttonValue = Serial.parseInt();
rightbuttonValue = Serial.parseInt();
```

Figure 14: Parsing a complete serial string. The values are mapped into the appropriate range and are ready to be written to the outputs.

make 3.3 VDC the highest output voltage. This is a little unfortunate because we lose some resolution, but I rather doubt the digitizer in the remote is that great, so it's fine.

We know that the *analogWrite* function takes a value in the range of 0-255 and then translates that into a PWM signal that mimics an analog output from 0-5 VDC. This means that each bit is worth about 20 mV. We want a maximum output of 3.3 VDC, which means we will use the range 0-166 or so in an ideal world. I found a value of 174 worked best, but more on that later.

In the *setup* function (**Figure 13**), we start up a serial port at 115200 baud to talk with the Python program and set the pin modes. The pins replacing potentiometers are set to be outputs, but the button pins are set to be inputs. The buttons in this controller have pull-up resistors, meaning that the input voltage to the microcontroller is normally a logic high. When the button is depressed, the pin is pulled low, and then goes high again when the button is released.

We can emulate the open button by setting the pin to be an input. In this high input impedance mode, the pull-up resistor does its job and we do not need to worry about generating a 3.3 VDC signal. When we want to pull down the pin to simulate a button press, we just make sure the pin is set to low and switch it to an output. Switching back to an input simulates the button release.

In the main loop of the sketch, we check if there is any serial data available. If so, we start parsing it (**Figure 14**). I used the *parseInt* function that looks for an integer in a comma-delimited serial character string. The parsing is nested inside of the *map* function. Mapping is a handy way to convert numbers in a certain range to analogous numbers in another range.

In this case, the serial input values are limited to be inclusively in the range -500-500. We want those values to correspond to 0 and 174, respectively. The *map* function

Figure 15: This project was a fun weekend build that is still entertaining. Here, the author prepares to take off while testing a revised version of the flight software.

does exactly this. You'll notice that for some axes, we map -500-500 to 174-0, which means that the control signal was the opposite sign of what the controller expected. You can buzz out the controller to determine this, do it by trial-and-error, or just trust the code and look into it if you have problems on your setup.

We also read the button values which are simply a 0 or 1. Once we hit the end of the sentence denoted by the newline character (\n), we write each of the analog values out to the corresponding PWM output pin. With two *if-else* blocks, we check the button values and set the pin to be a low output if the button is pressed, or leave it as an input if the button is not pressed. Sure, we could get rid of using *map* and these blocks to speed things up, but again, I don't think the controller would see much difference.

Test Flight

After uploading the Arduino sketch and checking my joystick code (not forgetting to set the serial port as described above), I was sure that something would happen, I just wasn't sure what. I turned on the controller, turned on the Python script, and tried the arming sequence. To fly the CX-10, you must first cycle its throttle through the maximum and minimum movements. I did this, heard the confirmation beep from the controller, and tried to slowly throttle up the copter. It did take off, but went shooting across the room. Back to the bench.

I checked the output voltages from the Arduino and everything looked fine. Had I really misunderstood the controller to such an extent? I tried several more times, then it worked. Once. The next time, it didn't. These problems can be very difficult to track down sometimes. If we know what causes the behavior, we can isolate it, but this seemed to be random. Knowing it couldn't be, I set about trying to keep everything between each test as consistent as possible. Even things that shouldn't matter.

After an hour or so of playing around, I finally figured out the issue. When started up and initialized, the controller expects the pitch, roll, and yaw controls to be centered. Why wouldn't they be? They are spring-loaded on the controller and we are just manipulating the throttle to arm the system. I had been leaving the joystick on the floor while setting everything up, with the stick just lying in whatever orientation. Centering the joystick, then turning on and arming the system gives consistently good results.

Now, back to that mysterious analog output value of 174 mentioned above. I experimented some with how well I could center the joystick on startup and determined that 174 gave me the best chance at getting an actual 3.3 VDC out at the maximum control movement with my rough centering. Going a little above at times hasn't seemed to hurt anything in the controller. I considered writing the *setup* function to force all outputs to center and go through the throttle arming sequence on startup, but I didn't like the idea of a self-arming drone, even if it is small. It would be too easy to accidently fly up into your face (that may sound like the voice of experience and indeed it may be).

Closing Thoughts

This project could go in several directions. For me, it is likely to be a toy and curiosity when my tech-minded friends and their kids are visiting. You can write Python programs (or Arduino sketches for that matter) that try to fly the quad in a pre-programmed path. With the continual need for pilot correction at the slightest gust, it may be quite a challenge. (Try flying in front of the HVAC vent in your house if you don't believe me!)

This same technique can be applied to all kinds of inexpensive toys, including RC cars, boats, or whatever else you can think of. It could also be fun to look into Alvaro's work on the controller protocol, making it possible to do away with the controller all together.

In the end, I was really happy with how this project came out (**Figure 15**). I don't know if I will put it in a nice case or make a permanent installation. That generally prevents me from continuing to play with the operation, and makes me think of it as a finished project. So, having this stay in its raw prototype form means that some Saturday evening, it could turn into an impromptu experimentation session.

Until next month, fly safely. **SV**

 Continued from page 19

Machine Vision Cameras

MIKROTRON GmbH introduces its new line of extremely compact high resolution/high speed CoaXPress® and CameraLink machine vision cameras.

All three new camera models, namely the EoSens® 25CXP+, EoSens® 12CXP+, and EoSens® 25CL+ are powered by an advanced OnSemi PYTHON CMOS sensor. The high resolution of the CMOS global shutter camera models combined with a precise trigger and synchronization, a high frame rate, and short integration times gives outstanding image quality and allows, for example, capturing even the smallest details of components in PCB (printed circuit board) assembly in high speed.

The extreme photo sensitivity of the high speed cameras of 5.8 V/lux*s @ 550 nm delivers reliable image information even in low light conditions. All camera models are based on a unique, robust, and compact design (80 x 80 x 66 mm) and are engineered for use in rough environments. The fanless design guarantees vibration-free operation.

The CXP+ models come with a four-channel CXP-6 CoaXPress® V1.1 interface, transmitting data at speeds up to 25 Gigabits per second in real time, while the CL+ camera takes advantage of CameraLink® technology.

EoSens® 25CXP+ offers 80 frames per second at 5,120 x 5,120 pixel resolution. EoSens® 12CXP+ offers 165 frames per second at 4,096 x 3,072 pixel resolution. Windowing down the image resolution further increases the frame rate up to 765 frames per second for 1,024 x 768 pixel resolution. The EoSens® 25CXP+ offers up to 80 frames per second at the extremely high 5,120 x 5,120 pixel resolution.

These new camera models meet the requirements for the most demanding tasks in all areas of application. They are ideally suited for all classical machine vision and scientific applications that combine the demand for high resolution as well as high speed. All models of the new high resolution/high speed camera family are GenICam and EMVA 1288 compliant, enabling easy handling and integration.

For further information, please contact:

MIKROTRON GmbH www.mikrotron.de

by Tom Carroll
TWCarroll@aol.com

How do You Define a Robot?

Over the years, I have been asked numerous times to define exactly what a robot is. Back in the summer of 2010, I wrote an article for *SERVO* entitled, *What Is a Robot*. I really didn't differentiate between what was a robot and what was *not* a robot. If the particular automaton I was discussing was considered a robot, I tried to classify it amongst other 'robots' that were in existence six years ago.

Robot Definitions

According to the Robot Institute of America back in 1979, the definition of a robot is: *"A reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks."* This is the definition by an *industrial* robot organization as there were few other types of robots in those early days.

The term "reprogrammable" implies that not only does the robot have some sort of computer that can allow its series of 'manipulations' to be placed in memory for a series of tasks, but the memory can be reprogrammed for another series of tasks at a later time. It was meant to *move* materials, parts, tools, and specialized devices, but it itself was not a mobile robot.

Another definition states: *"A real or imaginary machine that is controlled by a computer and is often made to look like a human or animal. A machine that can do the work of a person and that works automatically or is controlled by a computer."*

The following robot definition combines several of the above statements: *"A mechanical device that sometimes resembles a human and is capable of performing a variety of often complex human tasks on command or by being programmed in advance. A machine or device that operates automatically or by remote control."*

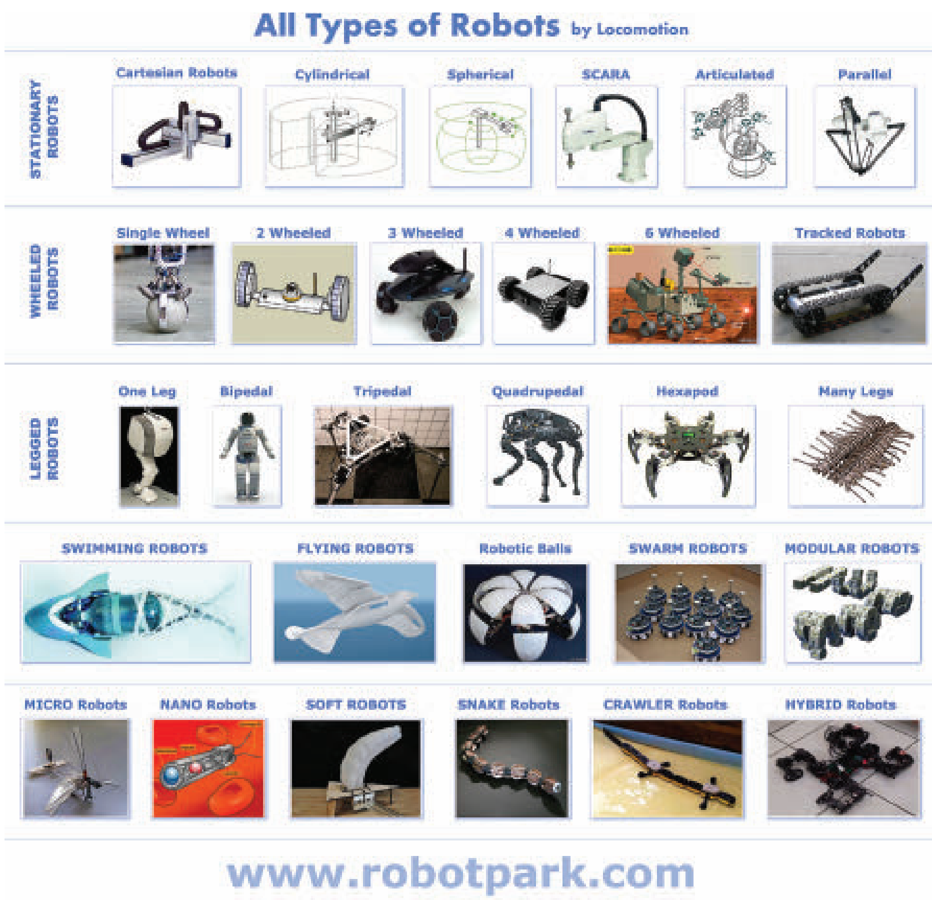


Figure 1. All types of robots. (Courtesy of RobotPark.com.)

To determine just what is considered a robot, I guess we can apply something like the duck test: "If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck." My version is: "If it looks like a robot, walks like a robot, and acts like a robot, then it is probably a who knows what."

I changed the last part for a good reason. There seems to be no clear definition of a robot. Complex mechanisms such as most robots seem to be are a bit more of a mystery to most people than an extremely common aquatic bird species. Robots take so many forms that just looking at a particular object that one thinks *looks* like a robot quite often is something entirely different. Early fictional robots were always bipedal humanoids.

I'll highlight a few widely different 'robots' as they are called by both the

Post comments on this article at **www.servomagazine.com/index.php/magazine/article/March2017_ThenandNow_Defining-Robots.**

public and those involved in the science of robotics. I'll start with how the name was derived and follow up with an author who popularized robots, then discuss a few movies that showed the creations of mankind in different ways. I'll also toss in a few unique robots that were the first of their kinds.

The chart in **Figure 1** illustrates a few of the many forms of what people call robots (courtesy of **robotpark.com**). Look closely at all of them and you'll see that many would have never made the cut according to the early definitions of robots mentioned previously.

The Word Robot is Born

The term 'robot' actually came from Czechoslovakian author, Karel Capek's 1920 play, *RUR* or Rossum's Universal Robots. The central characters in his play were artificial people called *roboti* and were created from synthetic organic matter at the RUR factory. I guess the term 'synthetic organic material' allowed Capek to write about a factory that synthesized something that looked like it was made from organic humanoid 'parts.' That certainly came in handy for the play to be able to use real humans to portray the central characters (as seen in **Figure 2**). They who had funny metal hats and shoulder pads to represent non-human creations.

Roboti is derived from the old Slavanic term, 'rabota' which means 'servitude' or slave labor. Karel and his brother, Josef changed the word to 'robot.' The robots were built to take over dull and repetitive human jobs. At the end of the play, these robots kill every human except for one who was viewed as not lazy.

Movie Robot Ethics

Robots have been used for almost a century to highlight ethical problems amongst human society. Problems with slavery, apartheid societies, labor unions, and societies with classes have used robots as characters to show human audiences another view of these subjects. When moving pictures were developed after the RUR play, it was easier to have elaborate sets with numerous human and robot characters that were not cost-effective for a stage play.

Figure 2. The robots of Rossum's Universal Robots.

The Robots of Metropolis

The great production from the late 1920s that used robots to portray a unique social problem is the classic movie, *Metropolis*. Produced in 1927 Germany, Director Fritz Lang had a young woman, Maria transformed into a *Maschinenmensch* (shown in **Figure 3***)* which is a metallic automaton shaped like a woman.

The transformation process is actually shown later in **Figure 4** with lighted rings causing the metamorphosis. Maria — originally a poor worker in the city's lower catacombs — was a part of a rebellion against wealthy industrialists who lived above in high-rise towers. This was the first actor to portray a mechanical humanoid in a robot suit.

Actress Brigitte Helm's robot suit was very uncomfortable to wear as it cut and bruised her, though Director Lang insisted that she had to wear it instead of using a stunt double. During the transformation scene, Helm actually fainted because the shot took so long that she couldn't get enough air in the restricting costume.

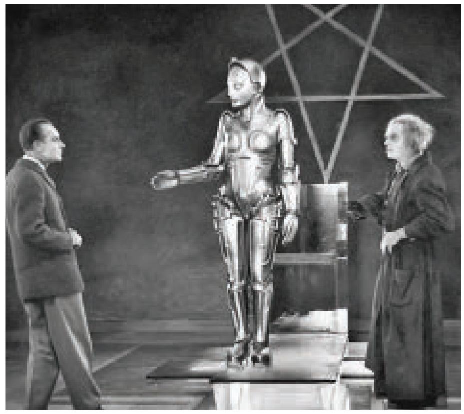


Figure 3. Maria in Fritz Lang's 1927 play, Metropolis.

Figure 4. Maria from Metropolis and C-3PO from Star Wars have some similarities.

The Twonky

The 1953 black and white comedy SciFi film, *The Twonky* was a box office bomb, but, strangely enough, it was planned to be a failure by its producer who needed a tax write-off for the year. I had read the initial short story decades ago when the Twonky was a 'hi-fi record player' constructed by an alien stuck on Earth inside the record player's factory.

The new plot had a college philosophy teacher purchasing his first television set while his wife was off on a trip.

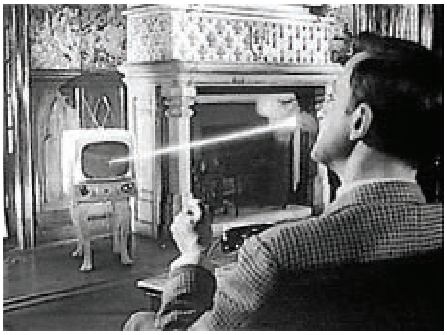


Figure 5. The Twonky lights the professor's cigarette.

It turns out that the TV shown in **Figure 5** was a bit more than a television when the protagonist, Kerry West discovers the set shooting a beam to light a cigarette as he is about to light it himself. Not really believing what he just saw, West is later convinced that the television is not normal when it also lights his pipe. Later, he finds that the TV can walk about on its legs, washes dishes, and vacuums the floor. It later manufactures a bunch of five dollar bills to pay the TV delivery man, allows him only one cup of coffee, and changes his choice of classical music to military marches.

The college football coach realizes that the Twonky seems to be serving only West when he kicked the professor as a test and the coach's leg was paralyzed. It later changes West's thoughts and paralyzes the football team. West is determined to get rid of the strange TV set, which resists his efforts, and the Twonky is later destroyed in a car wreck when it tries to control the driver. Definitely not your ordinary television set, or even robot.

More Ethics and Laws Become Part of Robotics

These earliest science fiction films and the play illustrate several very important features and philosophical issues that have driven key consequences in subsequent SciFi movies. The plots in *RUR* and *Metropolis* brought forth the premise that robots could not be trusted and their ultimate goal is to end mankind and take over in their place. Later films like *The Twonky* and other literature show robots in a different light.

Robots were portrayed by actors wearing robot suits in many movies before CGI became the norm, including those *Star Wars* favorites, R2D2 and C-3PO. George Lucas must have had inspiration from Maria of *Metropolis* when he envisioned C-3P0 for *Star Wars* as shown in **Figure 4**. C-3PO with his proper British butler personality became one of the most influential robot characters in movie history. His sidekick, R2D2 seemed more like a playful trashcan — a unique turn away from bipedal humanoid robots.

Robbie from the 1956 film, *Forbidden Planet* is another very memorable movie robot. In the scene shown in **Figure 6**, Robbie is told to fire the blaster at the ship's captain in a 'safety' demonstration since his robot brain will not allow killing of a human. He was extremely powerful, as are almost all movie robots — much beyond the capabilities of any actual robots of today. Fortunately, robots are now often portrayed as good guys.

Isaac Asimov's Influence on Robots, Fact, and Fiction

After Isaac Asimov's many robot stories showing how robots had a built-in aversion to hurting or killing a human being, robots in several movies became benevolent. Yes, the *Terminator* series had a few villainous humanoids, and even the swarm of NS-5 robots that wanted to take over Chicago in the Will Smith film, *I, Robot* were not mankind's best friends. However, more movies tried to show robots as a bit nicer. The scene in **Figure 6** is an example of how movies and literature made the turn from robot's malevolent actions towards man to benevolence.

In Asimov's 1942 short story, "Runaround," the three laws below are quoted from the *Handbook of Robotics*, 56th Edition, 2058:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Asimov was a prolific writer, having written over 500 books on all sorts of subjects — fiction and non-fiction. Asimov was born in Russia but grew up in New York, where he got his education. After completing his doctorate in biochemistry at Columbia University, Asimov joined the faculty of the Boston University School of Medicine where he remained for years. He became a full professor of biochemistry in 1979, but his greatest earnings were from his writings.

I find it a bit unique that he wrote so many stories about robots, given that his education was in biochemistry. If you carefully read them, he completely ignored any of the mechanical or electronic aspects

Figure 6. Robbie passes the Three Laws of Robotics test in Forbidden Planet.

of his robot characters except for the robot's 'positronic' brains (whatever *that* is). I briefly spoke with him while on a business trip back east years ago, and he was a very friendly and enthusiastic conversationalist, but viewed robots just as he would another person. He did not understand the engineering aspects of robots, though he wrote about all of the sciences in his many books.

It was Asimov's robot's adherence to the "Three Laws of Robotics" and the subsequent *psychological* aspects that were the center of his stories. One of his key characters was Dr. Susan Calvin, a robo-psychologist employed by US Robot and Mechanical Men, Inc., who seemed to drive many of the stories.

Early Military Robot Use in World War II

Figure 7 shows British soldiers looking at three captured Nazi Germany remote controlled tanks, called the Goliath tracked mine or *Leichter Ladungsträger Goliath*. Also called the Beetle Tank by the Allies, the initial model weighed 820 pounds and carried a 130 pound explosive charge to demolish tanks, buildings, and bridges. A second gas-powered model weighed 950 pounds and carried a 220 pound charge for up to 7.5 miles from the controller's station.

Built and used in 1942 to 1945, these electrically-driven mini tanks were controlled via a cable to a distance of up to a mile, or 7.5 miles for the newer model. Over 7,500 Goliaths were made but their success was low due to the very high unit cost and low ground clearance that stranded many of the vehicles. The dragged cable could be severed, rendering them ineffective. Many books and articles refer to these early war weapons as 'robot tanks.'

Grey Walter's Early Research Robots

The many movie robots that we've seen over the years certainly are interesting, and we secretly wish that most of them had the capabilities portrayed on the screen. Unfortunately, most robots today do not actually possess any sort of real intelligence ... or do they? If you had made that statement to Grey Walter, a British neurophysiologist back in the late 1940s, he would have convinced you that his simple robots with the equivalent of one or two brain cells could actually learn.

Figure 7. A few of the 1942 German Goliath tracked mines.

Electronics technology was in the analog stage at that time and vacuum tubes (or valves, as they are called in the UK) were the basis of electronic circuits. Walter — who was born in the US in 1910 but moved to the UK with his parents when he was five — had a life-long interest in the physiology of the brain, and actually improved upon the design of the electroencephalograph, or EEG that measures brain waves such as the alpha and beta waves. He used some of his background during WWII for guided missiles and radar systems. However, his most noted work was the development of autonomous robots.

Walter's ELSIE from 1948 — a machina speculatrix — is shown in **Figure 8**. Note the large (and heavy) lead-acid storage battery on the back, the two vacuum tubes in the center, and the steering motor just in front of the tubes. The tall photocell assembly sits on the axis of the front steered wheel and rotated along with the wheel. The front wheel was also driven, though the drive motor is on the other side of the wheel and not visible. This is a robot configuration that some experimenters should try instead of the more popular differential drive, as it has some qualities to allow better path and line following.

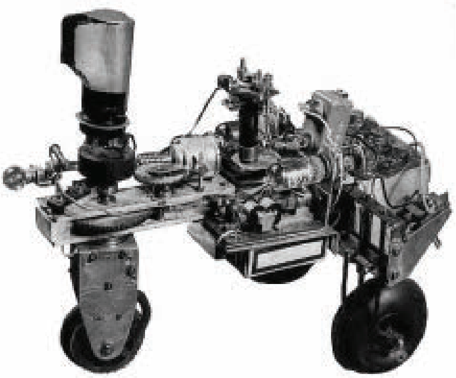


Figure 8. Dr. William Grey Walter's Elsie, a machina speculatrix.

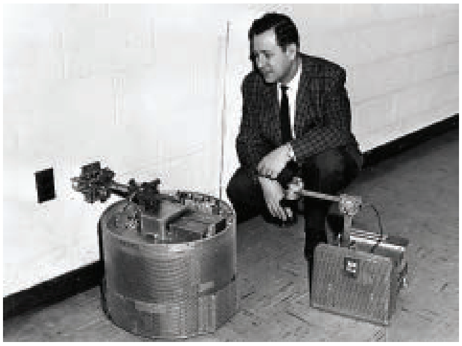


Figure 9. Johns Hopkins Beast and the diminutive Ferdinand are watched by Leonard Scheer.

The Hopkins Beast of the 1960s

In the early 1960s, staff and students of the Applied Physics Lab of the Johns Hopkins University built the larger Mod II robot called the Beast (shown in **Figure 9** with Leonard Scheer), and the smaller Mod 1, Ferdinand (on the right). These robots were designed to roam hallways, searching for wall outlets from which to 'eat,' or better stated, to recharge their batteries when their charge levels were low.

Using wall sensing and ranging ultrasonic sonars, power outlet-seeking photocell optics, and a wall plate-feeling arm, these robots accomplished all this without the use of computers. They were much more complex than Walter's Elsie. The

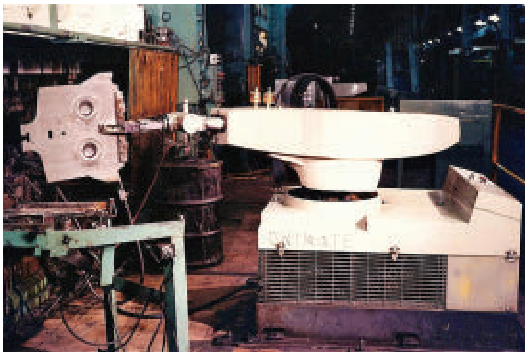


Figure 10. Early Unimate robot removing a hot forging from a press.

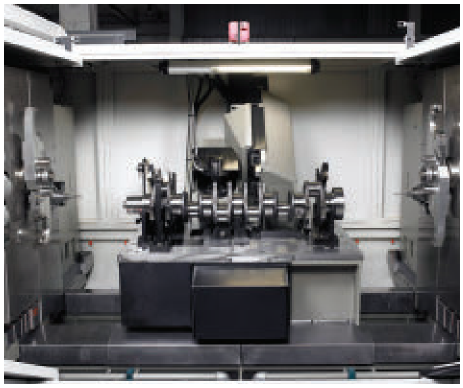


Figure 11. Caorle multi-axis machining system.

Figure 12. Unimate robot in a staged demonstration.

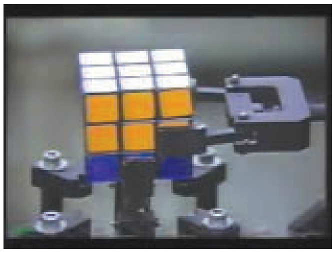


Figure 13. The Battelle Cubot's two Rubik's Cube grippers.

Beast's deliberate coordinated actions can be compared to the bacteria-hunting behaviors of large cells like a paramecium or an amoeba. Of course, microprocessors and microcontrollers were still a decade away from reality, so discrete transistor logic accomplished these motions. These days, the term robot is very popular, and so many types are now in existence. However, what allows one complex machine to be called a robot while another similar mechanical device is not called a robot?

Why are drones with four rotors flying around these days called robots and not radio controlled model helicopters? Why is the simple Unimation Unimate shown in **Figure 10** called a robot, and the complex multi-axis machining system shown in **Figure 11** not called a robot?

The Unimation Unimate

The Unimate was the first robot to be used in industry. In 1959, a Unimate similar to the one shown in **Figure 10** was installed on an assembly line at a General Motors plant in Trenton, NJ. The 2,700 pound Unimate #001 prototype is now a museum piece. By 1961, the Unimate 1900 series became the first mass-produced robot used for factory automation. Within a year, approximately 450 Unimate robots were employed in parts handling — especially hot castings and other dangerous manufacturing operations.

The scene in **Figure 12** is an example of how people began to stretch their imaginations in the early 1960s as to the capabilities of the new-fangled robots that were making the headlines. Rather than showing a boring demonstration of an industrial robot that can remove a hot forging from a huge factory press, why not use the Unimate 970 robot to pour a cup of tea for an attractive girl? "We want to let the world know that robots can do anything that a person can do," the robot industry touted.

In 1961, Joe Engelberger established Unimation, Inc., in Danbury, CT. Engelberger is often called the 'Father of Robotics' for founding the first industrial robot manufacturing company along with his partner, George Devol who formulated the idea and secured the patent for "Programmed Article Transfer." Soon after, their idea of robots in factories brought forth hundreds of companies with their own robotic solution products for industrial manufacturing industries worldwide. By the late 1970s, many of the companies that had their start in the US soon found themselves purchased by one of the many Japanese robot companies.

Battelle's Rubik's Cube-Solving Robot in 1982

We all know that robots are far more varied and useful than these first robots that I've written about. Today's machines take on so many forms. Using multi-axis mechanisms to manipulate the very popular puzzle, Rubik's Cube has been the goal of robotics engineers for over a quarter century. I remember a small robot built by Battelle Memorial Laboratories engineers, Phil Bondurant and Bob Dyer that was demonstrated by Dyer at our Robotics Society of Southern California meeting in early 1982.

Figure 13 shows a close-up view of the two grippers in the original Cubot mounted at 90° to each other. A third axis was not required to manipulate the 3x3x3 cube in order to arrange sides to be all one color. The robot was as big as a large suitcase and weighed over 70 pounds, but was compact enough for Dyer to take it around the country on demonstration tours.

It was not necessarily the manipulation of the cube that amazed me, but how the color camera could inspect all six sides, memorize the patterns, and then perform the necessary rotations in order to solve the puzzle. In the beginning, the robot could solve the puzzle in under five minutes, but further tweaking of the software brought the time under two minutes.

Sub1 Reloaded Breaks Rubik Cube-Solving Record

Figure 14. Sub1 Reloaded robot beat the Rubik's Cube-solving record at 0.637 seconds.

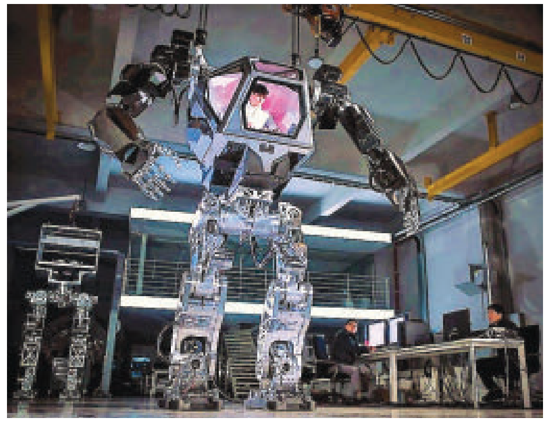


Figure 15. South Korean 13 ft tall Method 2 exoskeleton.

Back on November 12, 2016 — according to the World Record Academy — "Munich, Germany: A robot known as 'Sub1 Reloaded' took just 637 thousandths of a second to analyze the Rubik's cube and make 21 moves, so that each of the cube's sides showed a single color, thus setting the new world record for being the fastest robot to solve a Rubik's Cube."

The Sub1 Reloaded robot shown in **Figure 14** appeared at the Electronica Trade Fair and beat a previous record of 0.887 seconds. We're talking a fraction of a second here for both records. The timing started when shutters hiding the cameras were lifted and the system started. The record attempts used the same six high speed stepper motors and framework, but a different processor. The German company, Infineon provided its controller chip for the Sub1 Reloaded to highlight advancements in self-driving car tech, as well as automotive and industrial control systems.

When the 'shutters' covering the cameras were removed and the robot could 'see' the scrambled cubes and transmit that data to the processor, it, in turn, determined a solution and transmitted commands to six stepper motors. Each of the six steppers acted in pairs and held the cube via holes in the corners and spun the puzzle around to solve it. All of this was accomplished in a fraction of a second, and the number of moves were counted and displayed on the readout. The human record for this accomplishment is by a young boy in 2016 at 4.74 seconds. Not too bad for a human or a robot.

The Advancement of the Science of Robotics

Just think about how much progress was made between the two different robot systems that were used to visually analyze a Rubik's Cube initial configuration and then solve the puzzle. Stepper motors were implemented in both as they are a proven solution for rapid and precise rotary movement, though the steps were always 90°.

Cameras have progressed from vidicons to high-res CCDs, and discrete power transistors used in the Battelle robot have been replaced by MOSFETs and ICs. Computing power is now available from Raspberry Pi and Arduino controllers that are far cheaper and far more applicable to simple vision analysis and motor control.

Final Thoughts

Robots today take on so many different forms. It is even harder to determine just what *is* a robot and what is *not*. Complex mechanisms and automatons that we would not have called robots are all now grouped in that category.

The World War II tracked mines would now fall under the robot classification as do the remotely controlled quadcopters that people call drones. Motorized exoskeletons as well as motorized prosthetic arms and legs are now called robots. It seems almost all SciFi movies now have to have at least one robot, be it a helper for a human or an embedded computer within a spacecraft.

I'll leave you with one final automaton that we all can truly call a robot, though it is actually controlled by a human who sits inside. The huge Method 2 'mech' suit shown in **Figure 15** was built by master designer, Vitaly Bulgarov at the Hankook Mirae Technology labs in South Korea, and is truly a SciFi marvel that has come to life. It is supported by cables and has an overhead crane just as a safety measure. Who wants a multi-million dollar robot to crash on its face?

I'm guessing that one or more of you out there will build a robot similar to Method 2 one day. **SV**

The College of New Jersey Hosts
The 42nd ANNUAL ORIGINAL PERSONAL COMPUTER FESTIVAL

TCF'17™

NJ Makers Day Partner

Saturday, March 18, 2017 at The College of New Jersey, Ewing NJ

Focus on

Autonomous Vehicles

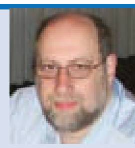
Keynote Speaker: Raj Rajkumar,
Professor of Electrical & Computer Engineering and Robotics Institute,
Carnegie Mellon University

Prof. Raj Rajkumar is the George Westinghouse Professor of Electrical & Computer Engineering and Robotics Institute at Carnegie Mellon University. He directs a number of labs at CMU, including the newly established University Transportation Center on Technologies for Safe and Efficient Transportation.

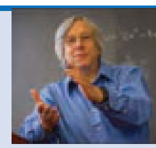
Featured Speaker: Greg Olsen,
Private Citizen Astronaut & Entrepreneur
" His Experiences Driving a Tesla with Autonomous Control"

Admission: $12.00 (advance)
$15.00 at Door
Free Parking / Wi-Fi
www.tcf-nj.org

at The College of New Jersey, Ewing, NJ
Saturday, March 18, 2017
9:00 am — 5:00 pm
Registration/Flea Market open at 9 am,Talks start at 10:15 am.



David Soll on Windows 10 and Microsoft Office



Author Barry Burd Speaks on Functional Programming and Developing for Bluetooth Beacons

INDOOR FLEA MARKET & VENDOR FAIR

50+ Talks, Workshops, Tutorials, Demos and Special Events!

Object-Oriented Programming University: One Day of Short Courses

Microsoft Presentations and Special Workshop on "Safe Computing"

Drone/Quadcopter/Maker Demos

Tour the Sarnoff Museum of Tech.

DAVID SOLL ON WINDOWS 10

Multimedia, Security, Wearable Technology, Games, Home Control and Historic Computers

Ham Cram Session & Exam: Get an Amateur Radio License in one day at TCF! Cram begins at 9 am.

Hands-on Arduino Workshop for Beginners

Raspberry Pi Tutorial and Projects

TCF Saturday Banquet with Door Prizes.
See TCF Web Site to Buy Tickets.

IEEE Integrated STEM Education Conference (ISEC)**
Saturday, March 11 – 8:00 am — 5:00 pm
at Friend Center, Princeton University, NJ
Theme: Connecting the Dots
Info: http://ewh.ieee.org/conf/stem/

IEEE/ACM IT PROFESSIONAL CONFERENCE
Friday March 17, 2017, 8:30 am — 5:00pm**
and continues as part of TCF's Saturday Talks.
For Conference Info and Fees see
http://princetonacm.acm.org/tcfpro/

For additional TCF'17 info, directions and advance tickets:
www.tcf-nj.org

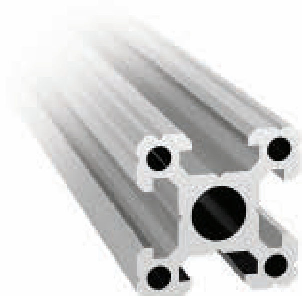
***The IT Pro (Friday) and ISEC Conferences require separate registrations.*

The 2017 TCF™ is sponsored by The College of New Jersey, its School of Engineering and supported organizations.

2017P-rev-08a

SERVOCITY.COM

WE HAVE THE PARTS FOR YOUR IDEAS



ACTOBOTICS
X-Rail

- Great for Creating Strong Lightweight Structures
- Awesome as a Linear Motion Guide
- Actobotics Pattern on the End

STANDARD & MINI
V-Wheels

- Built-in Bearings for Super Smooth Motion
- Use with a variety of standoffs &/or spacers
- Standard and Mini Versions

Visit ServoCity.com for the complete lineup of Actobotics parts!

ACTOBOTICS®

DREAM • DESIGN • BUILD • REPEAT

545400

Using a 585500 to attach a bottom tapped clamping mount

~ Weee

585506

545368 connected to 5/8" Tube

You can miter the edges

585494 (Beefy)

Dual Ball Bearing Hub (One of our faves!) 545444

Pattern Plate to Clamping Mount to HD Motor

Channel to X-Rail Roller Brackets to Standoffs to V-Wheels

585586

1" OD Tube to 1" ID Clamping Hub 545352

Channel Hard Mounted Directly to X-Rail

All of these ideas are just the tip of the iceburg! What will you make?

Actobotics X-Rail Fits inside a 1" ID Clamp such as 545464

585628

Mini V-Wheels for Compact Builds

Standard V-Wheels for More Strength