The Ultimate TABLEBot

By Camp L. Peavy, Jr.

Camp Peavy demonstrates his Phase III robot named "Buggy" at the October 2005 HBRC "TABLEBot Challenge".

At the April 2003 HomeBrew Robotics Club SIG (HBRC Special Interest Group) a few of us were lamenting the lack of robot building within the club. We noted building activity coincided with contests; specifically a recent line-following event and maze-busting contest called "the hurdle". We also noted that we had lots of table-top sized robots in the club including several Boe-Bots from a recent club buy. We thought in terms of what we wanted in a contest... we thought we'd like something with no setup… that is, we would like to use whatever was readily available. At the time we met in the "Castro Middle School" library and were surrounded by tables. What about table-top soccer? "Non-trivial" Bill Benson would say... well, we could put a net around the table to catch the robots should they fall... no… that would involve setup... that's when it was born... the TABLEBot!

The TABLEBot Challenge

Phase I: Build a robot that goes from one end of a table to the other and back.

Phase II: Have the robot push a block off the ledge of the table.

Phase III: Have the robot push the block into a shoebox mounted at the end of the table.

A "TABLEBot" is defined as a robot that survives, lives and plays on a table or pays the price. Now back to this non-trivial task of playing table-top soccer… like any big job or complicated process... you break it into smaller pieces or "phases" was the suggestion by Bill Hubbard. For the first phase we could simply have the robot go from one end of the table to the other. Then have the robot push a block off the ledge and finally have the robot push the block into a shoebox mounted at the end of the table. That's it! Those are all of the rules of the TABLEBot Challenge.

Dave Wyland anxiously watches his custom Boe-Bot as it completes "Phase I". Wyland's advice, "Never give Murphy and even break!"

There are no restrictions or limitations on the size or weight of the robot. Run what ya' brung! We don't even specify the size of the table, the block or how you mount the shoebox at the end of the table. A "TABLEBot" should simply be able to survive, live and play on a table... or pay the price. The TABLEBot Challenge rules are purposefully vague and non-restrictive so one can use whatever robot they have (Boe-Bot, SumoBot, LEGO…) and whatever table/block/box combination is readily available.

So it was decided that we would announce to the club at the official meeting the following week the "TABLEBot Challenge" and have "Phase I" for the June meeting, "Phase II" in August and "Phase III" in October. This would give builders two months between phases with a relatively simple beginning and increasingly difficult stages for August and October. It was emphasized by Wayne Gramlich that this was not a "contest" but a "challenge". Ever since it has been a joke within that club that anytime anyone refers to it as a "contest" they are immediately reprehended, "This is not a contest... it is a "challenge"." The non-competitive nature of the event makes for good-natured fun without competitive pressure, including the hassle of officiating. We treat it like "show and tell"; a regular feature at the end of each HBRC meeting. Participants are always allowed to show their creations in the best light.

The "ledge" itself represents the real excitement of the event… as relatively expensive robots hurl towards 30' high cliffs… 30' high to scale that is, if the robots were full-sized cars. The ledge keeps everyone on edge! Sometimes they fall; sometimes they break. This is the reason for 5-minute epoxy and this is why one of our club members, David Wyland, invented the "Wyland Leash"; a string tied around the robot to be held for debugging or if you're not completely confident of a particular environment (table color, lighting, etc). If you think about it this is really a practical exercise for a mobile robot. To paraphrase Clint Eastwood, "A robot's gotta' know its limits"… like say for instance the stairwell?

Ted Larson of "Ologic" demonstrates "Tracker" his color following TABLEBot.

In the October 2005 issue of SERVO magazine I introduced a Stamp based educational robot called the "PROTOBot" (Figure 1). Copies of the article are available at www.camppeavy.com/protobot.pdf. The original PROTOBot was built from common electronic components and radio-control (RC) airplane parts. It was an exercise in minimalism. I was trying to build an inexpensive but quality robot with as few parts as possible.

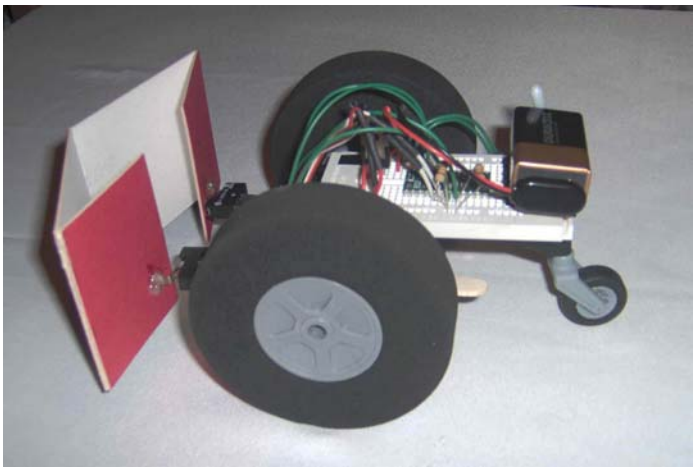


Figure 1: The PROTOBot: A Stamp-based educational robot.

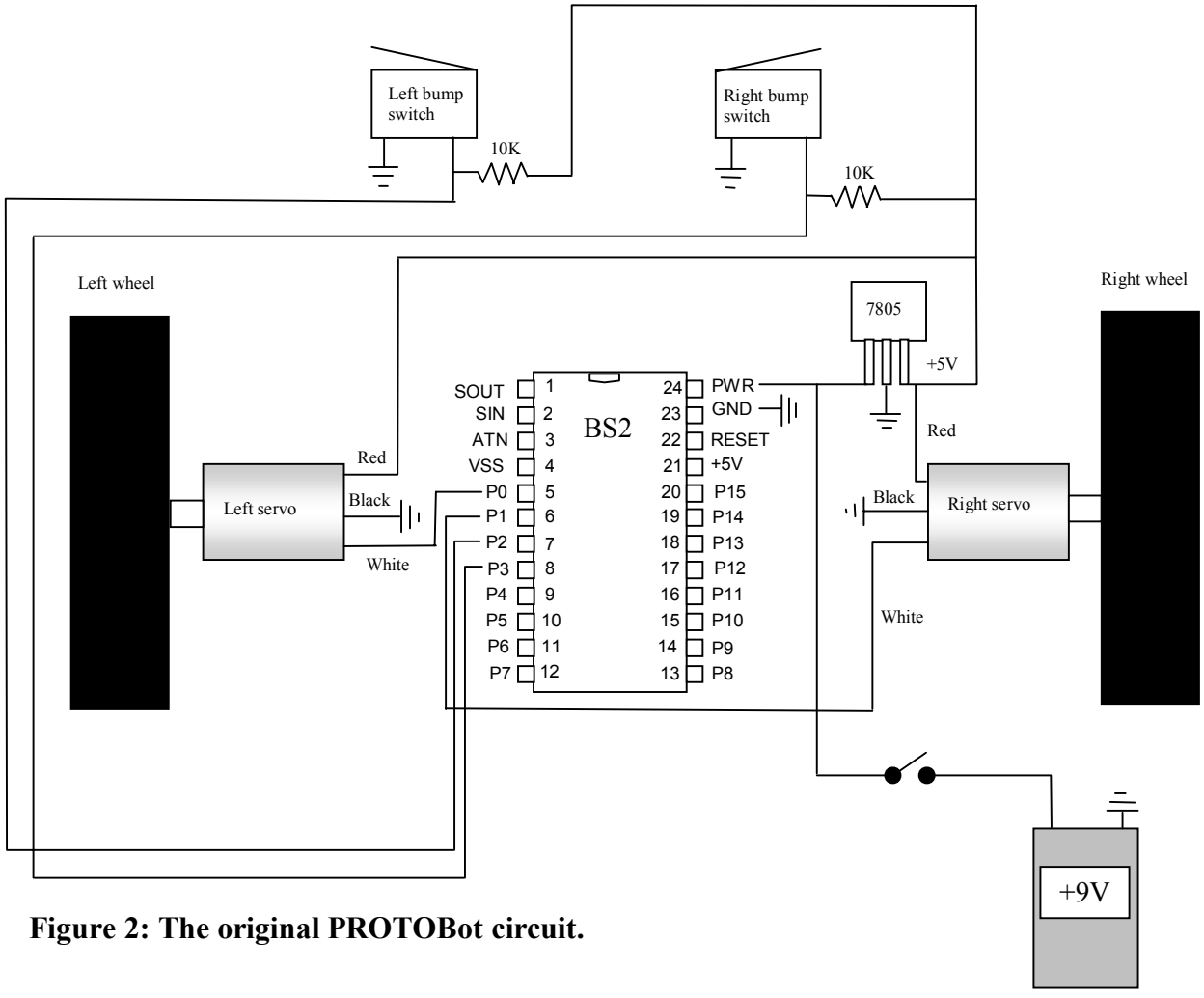


Figure 2: The original PROTOBot circuit.

```
' {$STAMP BS2}

x VAR Byte

fwd:
PULSOUT 0,1000
PAUSE 20
PULSOUT 1,500
PAUSE 20
IF IN2 = 0 THEN bwdr
IF IN3 = 0 THEN bwdl
GOTO fwd

bwdl:
FOR x = 1 TO 10
PULSOUT 0,500
PAUSE 20
PULSOUT 1,1000
PAUSE 20
NEXT
FOR x = 1 TO 3
PULSOUT 0,500
PAUSE 20
PULSOUT 1,500
PAUSE 20
NEXT
GOTO fwd

bwdr:
FOR x = 1 TO 10
PULSOUT 0,500
PAUSE 20
PULSOUT 1,1000
PAUSE 20
NEXT
FOR x = 1 TO 3
PULSOUT 0,1000
PAUSE 20
PULSOUT 1,1000
PAUSE 20
NEXT
GOTO fwd
```

Figure 3. BS2 Stamp program to make PROTOBot go forward; sense obstacles; backup and turn the opposite direction.

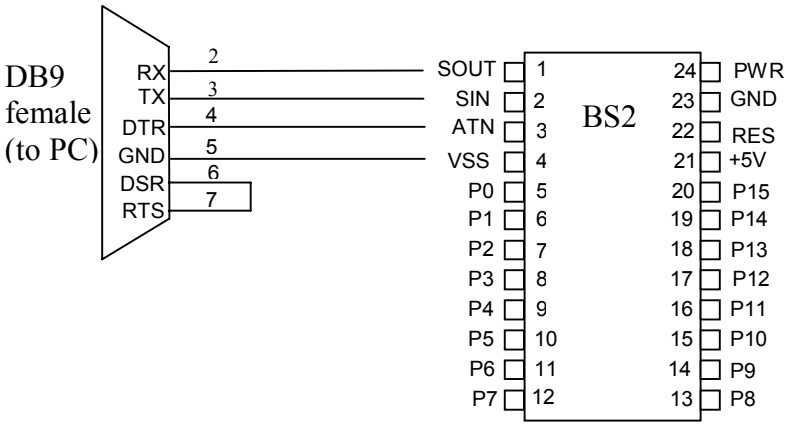


Figure 4: The programming cable connections from a PC to the BS2 Stamp.

In this article we will expand the PROTOBot into the Ultimate TABLEBot. We will add downward facing ledge sensors, a block-acquisition gripper, a dual rearward tablespace detector and more. In fact I want to use this project to totally pimp-out the original PROTOBot and use all 16 pins on the BASIC Stamp 2 (BS2) needed or not. The circuits in this article are from the Parallax manual or website with some modifications. The narrative describes my experiences for the edification of the community.

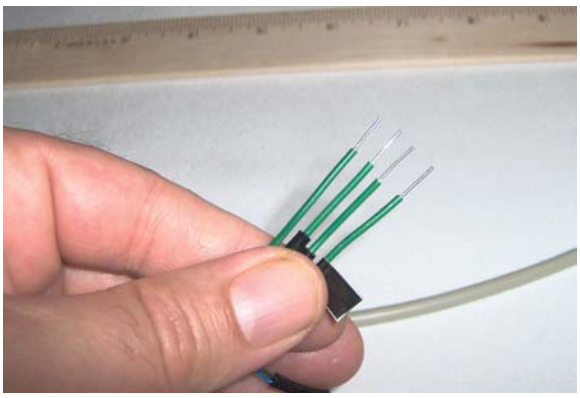


Figure 5: This is the homebrew plug. You may need to re-cut and re-strip the wires so that they are even. Tag or label the ground pin (5) with black tape and maintain the order of these 22-Gauge solid core wires as they will be plugged into pins 1-4 on the Stamp.

First let's take a look at the original PROTOBot circuit (Figure 2). Basically the PROTOBot is just a solderless breadboard on wheels controlled by a BS2 Stamp. The program we left off with in the previous article www.camppeavy.com/protobot.pdf was a module that made the robot go forward until it sensed an obstacle; at which point it would back up… turn the opposite direction and continue forward (Figure 3). We also built a homebrew cable (Figure 4) to program the Stamp by soldering 22 Gauge solid core wire to pins 2, 3, 4 and 5 and fashioning a plug to plug into the solderless breadboard (Figure 5).

Figure 6: E6000 industrial/craft adhesive. Right up there with Velcro, duct tape and 5-minute epoxy.

Much of the electronic construction of the PROTOBot involves simply soldering 22 gauge solid core wire to whatever you want to interface with the Stamp microcontroller. While professionally the solderless breadboard is considered a prototyping tool; for hobbyists it is often good enough.

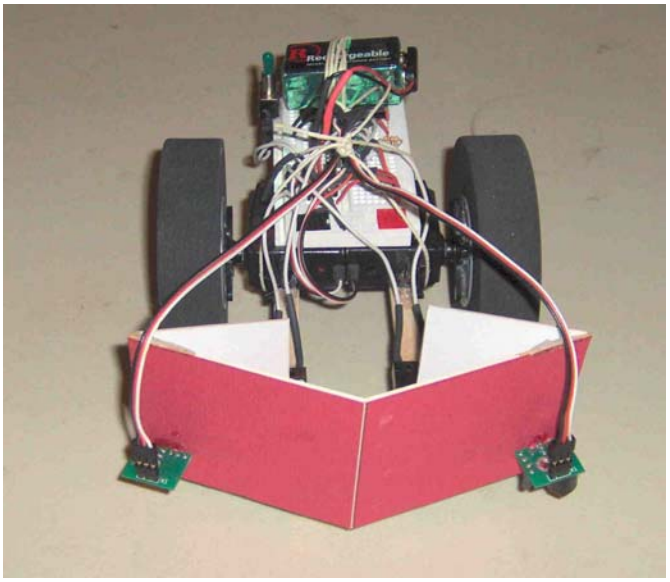


Figure 7: The first rule: stay on the table! E6000 (yes it's a verb) the IR detectors downward so to detect the ledge.

One of my favorite adhesives is E6000 because of its strength and shock absorbency (Figure 6). It is commonly available in hobby and craft stores. "Goop" and "Shoe-Goo" are similar but I really like the way E6000 sets. Another secret ingredient is 5-minute epoxy; not as shock absorbent as E6000 but cures in only 10 minutes! I made a point in the previous article that many good robots never get built because the builder doesn't have just the right screw or perhaps one never gets around to drilling a hole, etc, etc… just glue it… especially for a tabletop robot… chances are you're going move it anyway… so gluing gives you more options. E6000 is great for this. The mount is permanent but if you really want… you can peel it up, reposition and re-glue. In prototyping flexibility is a good thing.

The first thing to turn the PROTOBot into the Ultimate TABLEBot is something to detect the ledge of the table; after all "Phase I" involves simply going from one end of the table to the other and back. I've used a variety of sensors in this challenge including mechanical and ultrasonic but for this project I'm going to use the discontinued SSIR (Swanson Sensor IR) sensor from Parallax (Figure 7). The reason I am using the SSIR is I happen to have two of them. One of the great things about having built robots for a long time is you have plenty of spare parts (Figure 8).

Figure 8: The great thing about having built robots for a long time is plenty of spare parts… now finding them is another story.

It's basically an infrared LED and 38 kHz modulated receiver. The IR LED is modulated by the Stamp (FREQOUT PIN,1,38500) and the receiver pin goes low when it detects the 38kHz signal (Figure 9).

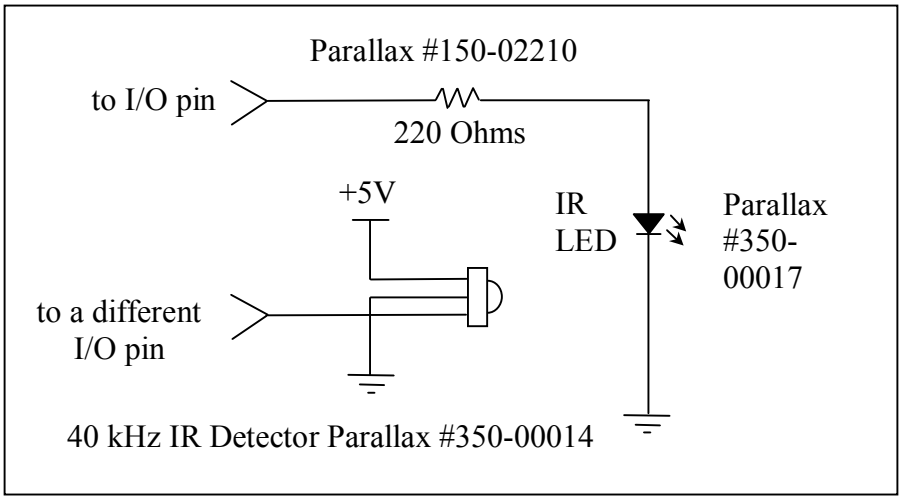


Figure 9: This is not the same circuit as the discontinued SSIR sensor but does detect IR strobed at 38kHz.

On my 2005 TABLEBot "Buggy" I used two "Ping" sensors from Parallax (Figure 10)… one low; one high. Basically the lower Ping looked for the *block* and the higher sensor looked for the *box*. I liked the strategy and have decided to do more of the same with this project except this time I mounted the top or higher Ping on a servo so it could swivel. This will help in centering the robot in relation to the *box* for Phase III (Figure 11) plus looks cool!

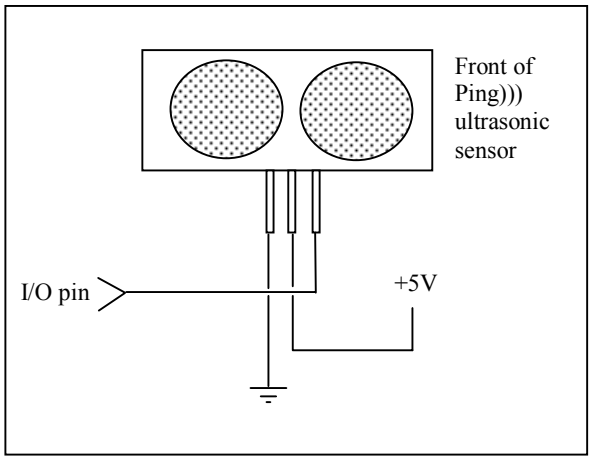


Figure 10: The Ping is as easy as it gets; pulsout/pulsin… one wire operation… the pulsin variable reads the raw time of flight.

My 2004 TABLEBot entry "Antsey" featured a servo actuated tail (Figure 12). Basically I mounted two micro switches on Popsicle sticks and before the robot reversed it would tap down its "tail" and verify "tablespace" before going backwards. Again it's great to have built robots for a while. Not only do you have plenty of spare parts but actual subassemblies. This time I borrowed Antsey's tail for… hey! Wait a minute… it's time this robot got a name!

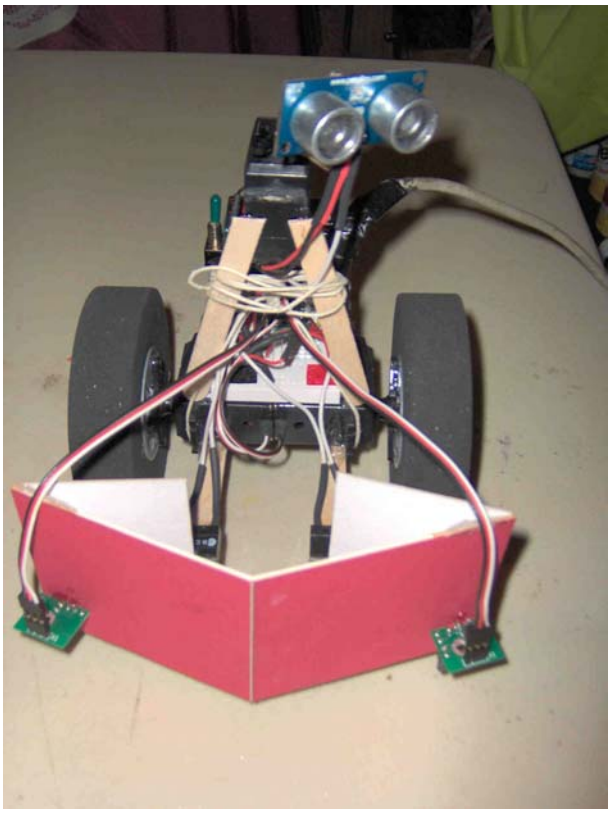


Figure 11: The high or upper Ping ultrasonic sensor swivels on a servo to find the box.

Naming your robot has got to be one of the real joys of robotics. Sometimes you start with a name and concept… other times it just comes to you as you work… this unit will be known as "Timmay" the Ultimate TABLEBot (Figure 13). I like it… it has rhythm… and personality!

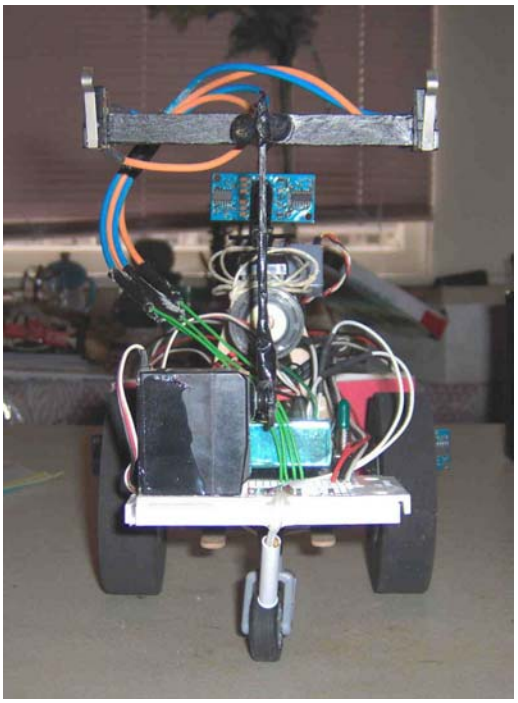


Figure 12: This is Timmay's rear view with the "tail" in the upward position. When the robot senses a ledge (front downward-facing IR sensors) the tail lowers and with the two microswitches tests for "tablespace" before reversing.

This allowed me to drive six servos from one pin on the Stamp. Now what to do with all those freed up pins… and what of the power situation? Everything worked fine - one at a time but when I tried to flex all the muscles at once the robot acted erratically.

Well duh, you're using one 7805 and those servos can draw .3A each… I found a neat trick that turned out I was just lucky. Of course the proper way to source more current is to use a bigger heatsink but I started reading on the web about paralleling voltage regulators… I tried it and it worked! Found out later I was just lucky in that the only reason it worked was probably due to the resistance in the solderless breadboard.

At first I tried to get away with only one ultrasonic sensor in the front but after working through the design felt I needed two; for differential sensing (Figure 14). The strategy is if the robot senses something on the left, turn right and if it senses something on the right, turn left. If it sees something in both; go forward! I've added a micro switch in the center of the bumper… when it gets depressed the "gripper" comes out and acquires the *block*. Once the *block* is acquired the higher Ping can focus on finding the *box*; into which the *block* will be deposited. In considering the necessity of detecting the table ledge I've elected to drive the servos from a "Scott Edwards" Servo Controller board (Figure 15)… again because I happen to have one. These boards are really cool for overcoming bandwidth limitations on the Stamp; especially with a mission critical function like staying on the table.

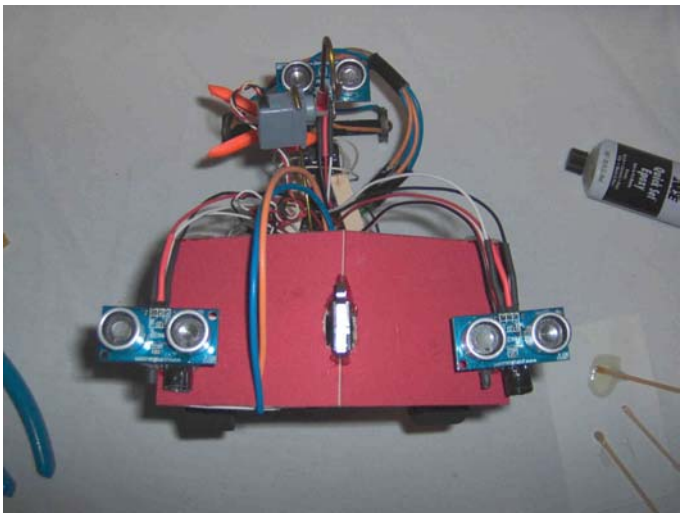


Figure 14: Here are the dual Ping ultrasonic sensors. They will be used to triangulate the location of the block. The center switch will detect whether the robot has acquired the block.

When thinking about programming robots put yourself in the robot's place. First the robot has to move; so pulse the wheels forward.

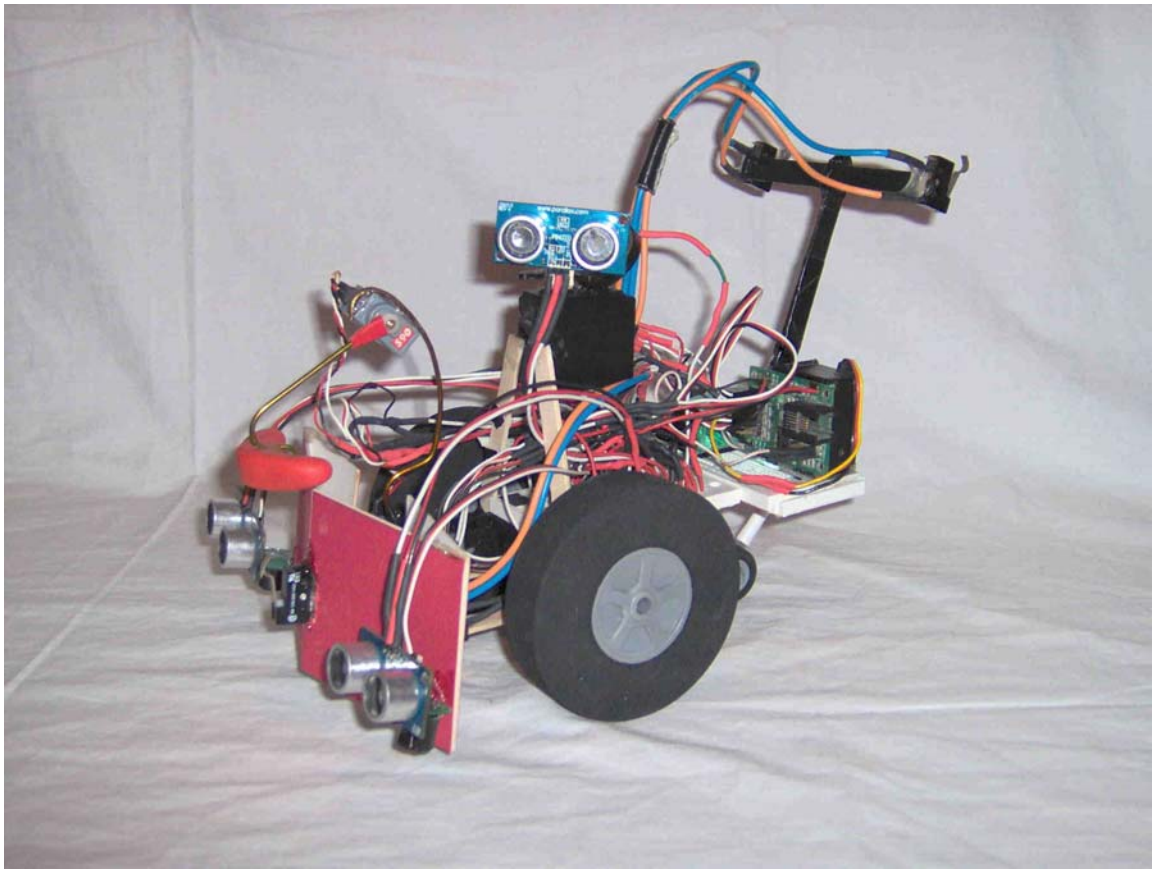


Figure 13: This is "Timmay" the Ultimate TABLEBot. He features differential drive 3.5" wheels, PROTOBot tri-bumper, dual downward facing IR sensors for detecting the ledge, forward facing differential Ping sensors for sensing the block; dual servo coat-hanger-wire arm and gripper w/ "Bake and Bend" Sculpy claw, palm switch for verifying block acquisition, Upper level swiveling Ping for finding the box; speaker for "beep-beep" voice, servo actuated dual rearward tablespace sensors and reed-switch/passive caster wheel based mobility detector. All controlled by a BS2 Stamp and Mini Serial Servo Controller (MSSCII). Rube Goldberg would be proud!

My goal with this robot was to use all 16 pins on the BS2 whether I needed them or not. It's both a totally pimped out PROTOBot and the Ultimate TABLEBot. I still have 3 more Stamp ports and two more available connections from the servo controller; front and rear CdS cells and a microphone come to mind.

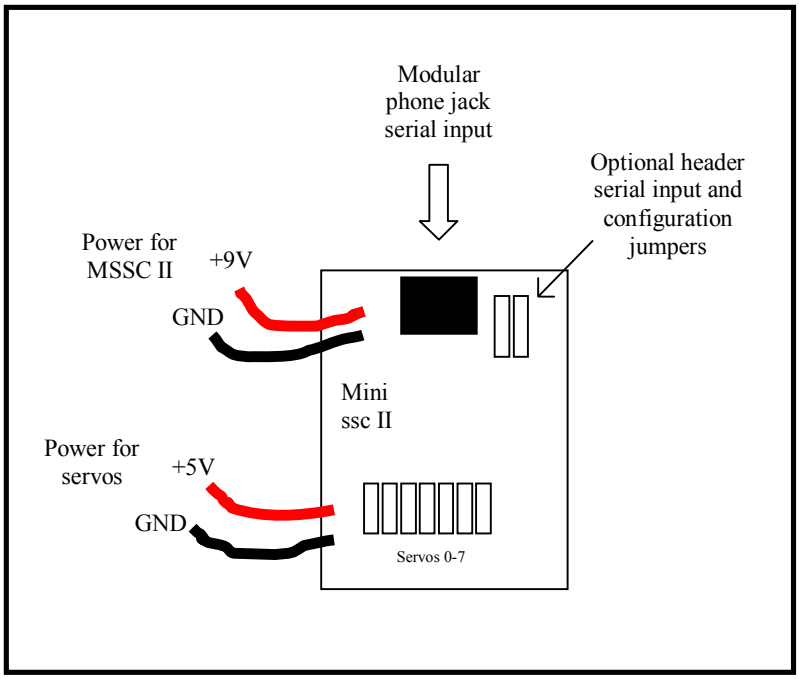


Figure 15: A specialized servo controller, like the Scott Edwards Mini SSC, helps overcome bandwidth limitations on the Stamp. Pulsing all those servos can overload the Stamp.

for the *box*. When the *box* is located move the *block* into *box* and you're done. The other sensors and output devices are icing on the cake; unnecessary but fun. All of the microswitch sensors use the standard switch interface (Figure 16). The speaker is useful for debugging without the "debug" command. Have it beep at different frequencies or intervals for different subroutines… this way you know what your robot's thinking (Figure 17).

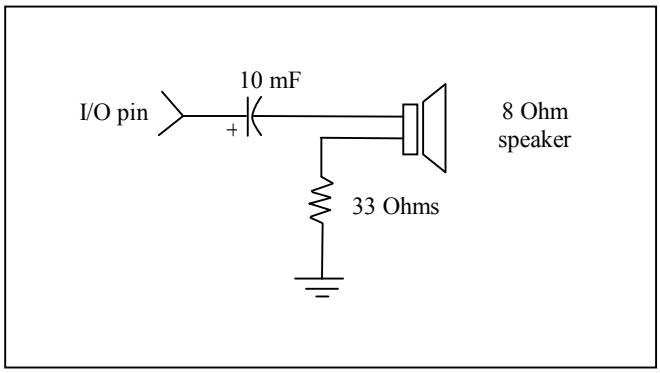


Figure 17: This is the basic speaker circuit. You could use a 40 Ohm speaker in place of the 8 Ohm and 33 Ohm resister. Give your robot a "beep-beep" voice.

Then you have to be sure that you DO NOT FALL OFF THE TABLE so look at your downward facing IR sensors. In addition you will want to "pulsout" the dual forward facing ultrasonic Ping sensors to try and find the *block*. When the robot detects the *block* on the left you want to turn right and when you detect the *block* on the right you want to turn left; when you detect the *block* on both Pings; go forward!
When the center switch (P11) gets pressed the robot assumes it's the *block* and actuates the gripper. When *block* has been acquired the high swivel Ping looks around

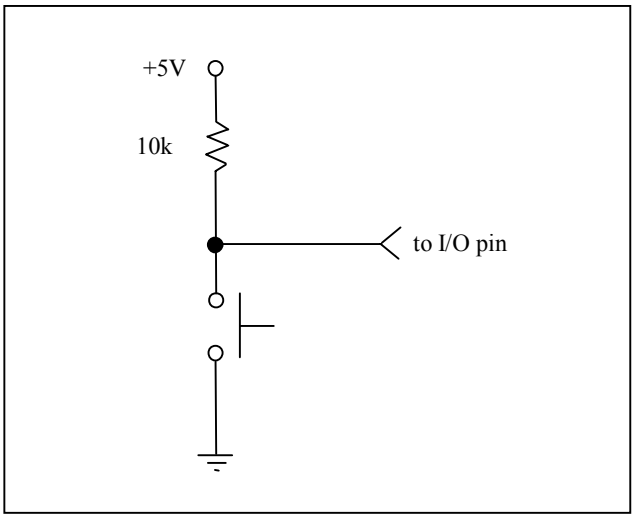
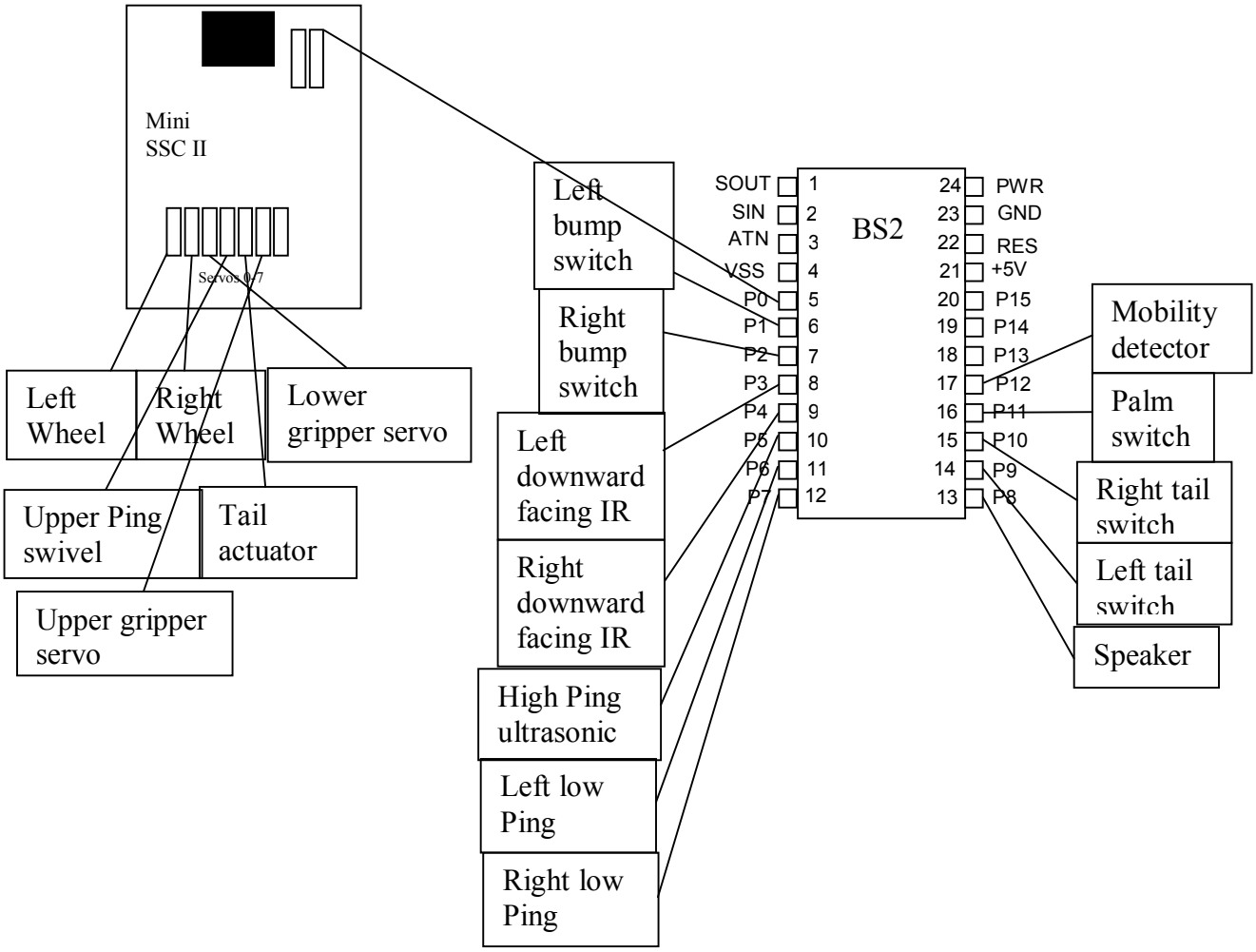


Figure 16: This is the circuit for the bump and tail switches. It is "active-low". That is when the button is pressed the TTL logic state goes to "0" or ground.

Here's a couple of tips for working with the BASIC Stamp. When you encounter weird problems check your power (Tip! Buy a battery tester). The Stamp will not run when connected to the computer unless you are running "debug". I assume



I'm using 12 of the 16 General Purpose I/O pins on the BS2 Stamp and 6 of the 7 servo ports on the Scott Edwards MiniSSCII. The program below accomplishes the TABLEBot Challenge "Phase III".

it has something to do with the DTR connection from the computer. Build and test, build and test, build and test. Make frequent incremental saves of your code; "Save As" and increment the number by one. Use debug commands as you debug the system and remark (') them out as you finish testing and especially before you place the robot on a "live" table. This way if you have to test again you can simply un-rem them. As you drop a small part look quickly where it's going; otherwise it might be lost forever. Find tools that you enjoy working with. It makes the activity more enjoyable.

In building robots, as in life, all things conspire to keep you from doing what needs to be done. If it's not one thing it's another (tip: turn off the TV). Regardless, remember that building is a series of small steps. Don't think about the complexity of the overall project or you will become discouraged. Do the small things on a daily basis and eventually the whole thing will be done. If you come to a dead-end or stopping point with one

subsystem (glue drying or need a part) consider what can be done on another subsystem… I think too many times we try and come up with reasons as to why things can't be done or we didn't get around to them rather then "just doing them" to paraphrase the Nike motto. Enter contests! There's nothing like a deadline to force one to create.

At this point Timmay is capable of depositing the block-in-the-box-on-a-table but I still haven't used all 16 pins on the BS2 Stamp plus I haven't integrated the "mobility detector"; a small rare earth magnet (Radio Shack #64-1895) embedded in the passive caster tire and a small reed switch (RS#49-496) glued to the solderless breadboard body. I pipe the On/Off signal from the reed switch into P12 and as long as the wheel is turning the robot is "happy". The good news is I still have until June for the 2006 TABLBot Challenge "Phase I" and October for "Phase III". This will be our 4th annual www.hbrobotics.org.

```
' {$STAMP BS2}

' Variables
x                VAR Byte    ' Generic counter variable
holdbit3         VAR Bit     ' Bit for left downward facing IR sensor.
holdbit4         VAR Bit     ' Bit for right downward facing IR sensor.
rawdata1         VAR Word    ' Variable for swivel Ping (upper)
rawdata2         VAR Word    ' Variable for left Ping (lower)
rawdata3         VAR Word    ' Variable for right Ping (lower)

main:            ' Main programing loop

' "Scott Edwards" Mini Serial Servo Controller (MSSCII) initial settings
SEROUT 0,$4054,[255,0,254]    ' Left drive wheel; Forward = 254, Stop 131
SEROUT 0,$4054,[255,1,0]      ' Right drive wheel; Forward = 0, Stop 130
SEROUT 0,$4054,[255,2,127]    ' Swivel Ping servo, Higher left
SEROUT 0,$4054,[255,3,254]    ' Tail servo, Higher up
SEROUT 0,$4054,[255,5,254]    ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,125]    ' Gripper low, Lower back (125,254)

'FREQOUT 8,1000,2500,250      ' Audio speaker

' Downward facing IR sensors
FREQOUT 3,1,38500       ' Left downward facing IR (table ledge sensor)
holdbit3 = IN3
'DEBUG "sensor = ",DEC holdbit3,CR

FREQOUT 4,1,38500       ' Right downward facing IR (table ledge sensor)
holdbit4 = IN4
'DEBUG "sensor = ",DEC holdbit4,CR

' IR Table Matrix
IF holdbit3 = 1 AND holdbit4 = 1 THEN filter   ' Backward and turn opposite direction
IF holdbit3 = 1 THEN filter                    ' Backward and turn right
IF holdbit4 = 1 THEN filter                    ' Backward and turn left

' Differential "Block" ultrasonic
IN6 = 0                     ' Set trigger
PULSOUT 6, 1                ' Activate sensor
```

```
PULSIN 6, 1, rawdata2              ' Measure echo pulse (Rawdata2 for left Ping!)
PULSOUT 5, 1                       ' Activate sensor
PULSIN 5, 1, rawdata1              ' Measure echo pulse (Rawdata2 for left Ping!)
'DEBUG ? rawdata1
'DEBUG ? rawdata2, CR              ' 5000 = ~3' / 600 = ~7" away
IF rawdata2 < 300 THEN block


IN7 = 0                            ' Set trigger
PULSOUT 7, 1                       ' Activate sensor
PULSIN 7, 1, rawdata3              ' Measure echo pulse (Rawdata3 for right Ping!)
'DEBUG ? rawdata3, CR              ' 5000 = ~3' / 600 = ~7" away
IF rawdata3 < 300 THEN block

'PAUSE 1000

' Center palm switch
'DEBUG ? IN11, CR                  ' Look at value in Pin 11 (Palm switch)
IF IN11 = 0 THEN grab              ' 0 = switch pressed / Grab block

' Front bumper switches
IF IN1 AND IN2 = 0 THEN bwdo       ' Backward and turn opposite direction
'DEBUG ? IN1, CR                   ' Look at value in Pin 1 (Left bump switch)
IF IN1 = 0 THEN bwdr               ' IF left bump then back and turn right
'DEBUG ? IN2, CR                   ' Look at value in Pin 2 (Right bump switch)
IF IN2 = 0 THEN bwdl               ' IF right bump then back and turn left

'PAUSE 1000
GOTO main                          ' Return to main programming main

bwdo:
SEROUT 0,$4054,[255,0,131]     ' Forward 254, Stop 131       'Stop!
SEROUT 0,$4054,[255,1,130]     ' Forward 0, Stop 130

SEROUT 0,$4054,[255,3,0]       ' Tail servo, Higher up       'Lower tail to check for tablespace
PAUSE 500
IF IN9 = 1 THEN halt           ' If no reverse tablespace do not back up
IF IN10 = 1 THEN halt          ' If no reverse tablespace do not back up

SEROUT 0,$4054,[255,3,254]     ' Tail servo, Higher up       'Bring tail back up if tablespace
PAUSE 500

FOR x = 1 TO 5
SEROUT 0,$4054,[255,0,0]       ' Forward 254, Stop 131       'Reverse
SEROUT 0,$4054,[255,1,254]     ' Forward 0, Stop 130
PAUSE 50
NEXT
SEROUT 0,$4054,[255,0,254]     ' Forward 254, Stop 131       'Turn right
SEROUT 0,$4054,[255,1,254]     ' Forward 0, Stop 130
PAUSE 1000
'DEBUG "Left and Right", CR
GOTO main

bwdr:
SEROUT 0,$4054,[255,0,131]     ' Forward 254, Stop 131       'Stop!
SEROUT 0,$4054,[255,1,130]     ' Forward 0, Stop 130
```

```
SEROUT 0,$4054,[255,3,0]     ' Tail servo, Higher up       'Lower tail to check for tablespace
PAUSE 500
IF IN9 = 1 THEN halt         ' If no reverse tablespace do not back up
IF IN10 = 1 THEN halt        ' If no reverse tablespace do not back up

SEROUT 0,$4054,[255,3,254]   ' Tail servo, Higher up       'Bring tail back up if tablespace
PAUSE 500

FOR x = 1 TO 5
SEROUT 0,$4054,[255,0,0]     ' Forward 254, Stop 131       'Reverse
SEROUT 0,$4054,[255,1,254]   ' Forward 0, Stop 130
PAUSE 50
NEXT
SEROUT 0,$4054,[255,0,254]   ' Forward 254, Stop 131       'Turn right
SEROUT 0,$4054,[255,1,254]   ' Forward 0, Stop 130
PAUSE 250
'DEBUG "Left", CR
GOTO main

bwdl:
SEROUT 0,$4054,[255,0,131]   ' Forward 254, Stop 131       'Stop!
SEROUT 0,$4054,[255,1,130]   ' Forward 0, Stop 130

SEROUT 0,$4054,[255,3,0]     ' Tail servo, Higher up       'Lower tail to check for tablespace
PAUSE 500
IF IN9 = 1 THEN halt         ' If no reverse tablespace do not back up
IF IN10 = 1 THEN halt        ' If no reverse tablespace do not back up

SEROUT 0,$4054,[255,3,254]   ' Tail servo, Higher up       'Bring tail back up if tablespace
PAUSE 500

FOR x = 1 TO 5
SEROUT 0,$4054,[255,0,0]     ' Forward 254, Stop 131       'Reverse
SEROUT 0,$4054,[255,1,254]   ' Forward 0, Stop 130
PAUSE 50
NEXT
SEROUT 0,$4054,[255,0,0]     ' Forward 254, Stop 131       'Turn left
SEROUT 0,$4054,[255,1,0]     ' Forward 0, Stop 130
PAUSE 250
'DEBUG "right", CR
GOTO main

grab:
SEROUT 0,$4054,[255,0,131]   ' Left drive wheel Forward = 254, Stop 131
SEROUT 0,$4054,[255,1,130]   ' Right drive wheel Forward = 0, Stop 130

SEROUT 0,$4054,[255,5,0]    ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,125]   ' Gripper low, Lower back (125,254)
PAUSE 500

SEROUT 0,$4054,[255,5,0]    ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,254]   ' Gripper low, Lower back (125,254)
PAUSE 500

SEROUT 0,$4054,[255,5,175]   ' Gripper high, Higher down (254,0,175)
```

```
SEROUT 0,$4054,[255,4,254]     ' Gripper low, Lower back (125,254)
PAUSE 500

IF IN11 = 0 THEN hold          ' If block sensed hold

SEROUT 0,$4054,[255,5,0]     ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,254]     ' Gripper low, Lower back (125,254)
PAUSE 500

SEROUT 0,$4054,[255,5,0]     ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,125]     ' Gripper low, Lower back (125,254)
PAUSE 500

SEROUT 0,$4054,[255,5,254]     ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,125]     ' Gripper low, Lower back (125,254)
PAUSE 500

'DEBUG "test", CR

GOTO main

hold:
IF IN11 = 1 THEN main
SEROUT 0,$4054,[255,5,175]     ' Gripper high, Higher down (254,0,175)
SEROUT 0,$4054,[255,4,254]     ' Gripper low, Lower back (125,254)
PAUSE 20

FREQOUT 8,1000,2500,250        ' Audio speaker

FOR x = 127 TO 254
SEROUT 0,$4054,[255,2,x]    ' Swivel servo, Higher left
NEXT

FOR x = 254 TO 127
SEROUT 0,$4054,[255,2,x]    ' Swivel servo, Higher left
NEXT

FOR x = 127 TO 0
SEROUT 0,$4054,[255,2,x]    ' Swivel servo, Higher left
NEXT

FOR x = 0 TO 127
SEROUT 0,$4054,[255,2,x]    ' Swivel servo, Higher left
NEXT
GOTO hold

halt:
SEROUT 0,$4054,[255,0,131]    ' Left drive wheel Forward = 254, Stop 131
SEROUT 0,$4054,[255,1,130]    ' Right drive wheel Forward = 0, Stop 130
'DEBUG ? IN9
'DEBUG ? IN10
IF IN9 AND IN10 = 0 THEN main
IF IN10 AND IN9 = 0 THEN main
GOTO halt

filter
```

```
'Downwa:rd facing IR sensors
FREQOUT 3,1,38500           ' Left downward facing IR (table ledge sensor)
holdbit3 = IN3
'DEBUG "sensor = ",DEC holdbit3,CR

FREQOUT 4,1,38500           ' Right downward facing IR (table ledge sensor)
holdbit4 = IN4
'DEBUG "sensor = ",DEC holdbit4,CR

'IR Table Matrix
IF holdbit3 = 1 AND holdbit4 = 1 THEN bwdo   ' Backward and turn opposite direction
IF holdbit3 = 1 THEN bwdr                    ' Backward and turn right
IF holdbit4 = 1 THEN bwdl                    ' Backward and turn left
GOTO main

' Timmay sees block
block:
'SEROUT 0,$4054,[255,0,131]    ' Left drive wheel Forward = 254, Stop 131
'SEROUT 0,$4054,[255,1,130]    ' Right drive wheel Forward = 0, Stop 130
'DEBUG ? IN9
'DEBUG ? IN10
IF IN11 = 0 THEN grab              ' 0 = switch pressed / Grab block
IF IN9 AND IN10 = 0 THEN main
IF IN10 AND IN9 = 0 THEN main

IN6 = 0                        ' Set trigger
PULSOUT 6, 1                       ' Activate sensor
PULSIN 6, 1, rawdata2              ' Measure echo pulse (Rawdata2 for left Ping!)
'DEBUG ? rawdata2, CR                ' 5000 = ~3' / 600 = ~7" away
'PAUSE 250

IN7 = 0                        ' Set trigger
PULSOUT 7, 1                       ' Activate sensor
PULSIN 7, 1, rawdata3              ' Measure echo pulse (Rawdata3 for right Ping!)
'DEBUG ? rawdata3, CR                ' 5000 = ~3' /600 = ~7'
'PAUSE 250

' Decision matrix
IF rawdata2 < 500 AND rawdata3 < 500 THEN main
IF rawdata2 > 500 AND rawdata3 > 500 THEN main
IF rawdata2 > rawdata3 THEN right
IF rawdata2 < rawdata3 THEN left
IF rawdata2 = rawdata3 THEN main
IF IN11 = 0 THEN grab              ' 0 = switch pressed / Grab block
GOTO block

right:
SEROUT 0,$4054,[255,0,254]    ' Forward 254, Stop 131       'Turn right
SEROUT 0,$4054,[255,1,130]    ' Forward 0, Stop 130
GOTO block

left:
SEROUT 0,$4054,[255,0,131]      ' Forward 254, Stop 131       'Turn left
SEROUT 0,$4054,[255,1,0]      ' Forward 0, Stop 130
GOTO block
```