Cruise Control For Live Steam Loco's

By

DeWinton Dave

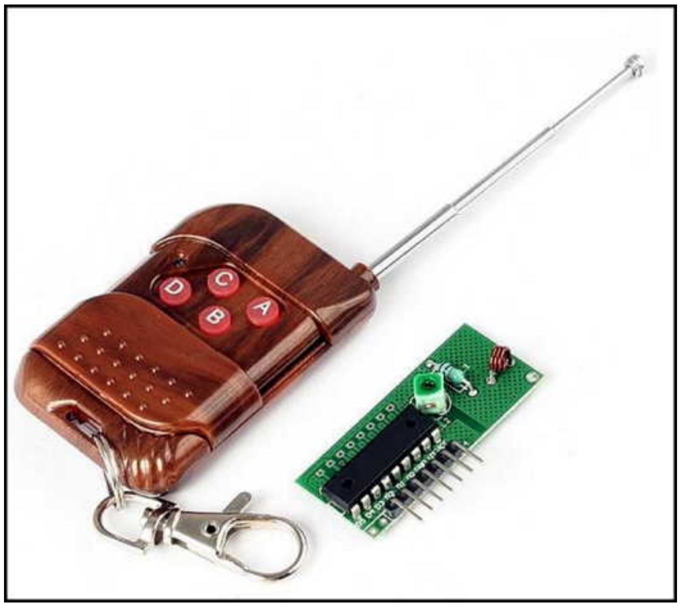
(Live Steam Forum/Topic: Cruise control for live steam loco)

(02 Jun 2012) - Originating Reply:
Here's the start of the build details for Dave's cruise-o-matic. A fellow G1MRA member inspired me to have a go at building cruise control for my live steamer. His controllers are based purely on electronics with no software. I don't have much electronics know how. Then I discovered the excellent Picaxe system **PICAXE - Home Page** and the rest is history...

The aforementioned G1MRA member also uses 433MHz radio control for his steamers. And I had a go at this as well. 433MHz is the garage door opener / wireless doorbell type frequency. Key fob size transmitters are freely available. I bought a Tx and Rx module from eBay recently, here is the link: **433MHz 4-Channel RF Wireless Controller + Remote**

This is the seller's link: **Kimke-Wei - eBay Store**

These are really good value with free worldwide shipping and the seller is excellent.



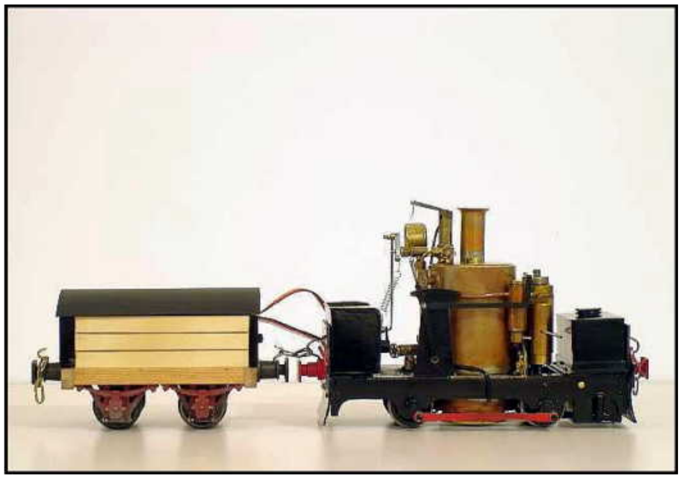
This is what the Tx and Rx look like, the transmitter even with classy wood grain effect finish.

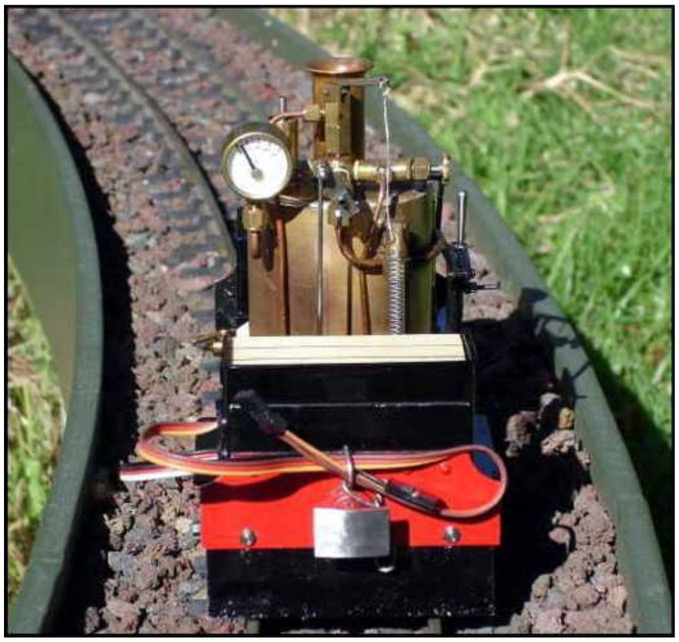
This system has 4 channels but I'm only using two at present. On my set-up, button D starts and stops the train and sounds the whistle at the same time. Pressing any other key will just sound the whistle.

The 433MHz system is very light on battery power compared to the usual model aircraft type controllers which are constantly transmitting instructions at a great rate, even when not appropriate. My 433MHz transmitter only transmits when a button is pressed, and the receiver only draws a few milliamps. As with the usual radio controllers current is being sent to the servos even when they are still, this is to maintain position. So I turn off the servos when —

they are not in use, like when the stop train button has been pressed and when the whistle is not being blown. The system is glitch-free and the range is adequate for my garden.

I have fitted the control system in a trailing truck, here is the rig:





There are two servos in the bunker; a mini servo on the bottom controlling the regulator and a micro servo glued on top of it, this pulls the whistle cord. I've used a spring in the whistle cord to give the whistle lever a full pull without straining anything.

Here is a video of the loco doing a circuit of our track; the first minute or so deals with the loco preparations:

Comment Kevin Strong:
"Cruise Control" generally means some means of maintaining a constant speed through variations in grade, etc. Is that the same as what we're discussing here?

(03 Jun 2012) - Reply:
Yes, that's what it does. It's set to maintain a scale 10 mph. You can see some of the control action in the above video after the first minute or so:

Comment Eric Schade:
That is pretty Cool Dave! I hope to see how you did it, perhaps even an article in Steam in the Garden? I might add a second whistle signal if it were mine but yours sounds great.

Comment Greg Elmassian:
Very interesting. How does it know how fast it is going? This is of course necessary to be able to maintain a constant speed under varying conditions.
What part does the PICAXE play? I'd guess you have a speed sensor of some time, and when it goes below a certain amount, the throttle is triggered until the speed comes up.
It does not seem to have over speed control, nor proportional control of the throttle.

Comment David Leech:
Dave: I know that one G1MRA member who has been active in this type of work for many years uses disks attached to the axle to calculate speed etc. and keep it constant.
I get the impression that yours is quite variable as you can hear the acceleration from time to time.
Is this something that you are still working on? It all seems very interesting, but I think that I will keep 'burning my fingers' for now!

(03 Jun 2012) - Reply:
Hi Greg: The trailing car has a counter on one axle; a Picaxe counts pulses over a period of time and outputs whether the count is high or low. Another Picaxe reads the others output and moves the regulator servo accordingly.

To avoid kangaroo hopping I have set the regulator to shut off to a constant position and then close the regulator by a single step every second that the speed is still high, this is hard to see. To keep over speed to a minimum during the speed up procedure the controller is now closing the regulator in one movement from however open it is. Every weekend I keep making changes to the software. It's very easy and satisfying. The loco has just 4 throttle opening positions during running. I have found for this loco that large throttle openings work best.

(03 Jun 2012) - Reply:
Hi David: I just use 4 throttle opening positions currently. The loco is only running slowly, I have found that fine increments won't counter a sudden slowing down. A delay before a subsequent opening of the throttle is required to allow the train to build up speed. I am constantly tweaking the settings.

I built the cruise control so that the loco could just run round and round the line at a slow speed without stalling.

(03 Jun 2012) - Reply:
Hi Eric: you could have what ever whistle sequence you like. I have a thing for the "peep-pip-pip" sound.

Comment Greg Elmassian:
Thanks for the explanation Dave. I'm curious why you need two processors; did you run out of inputs/outputs? I would not think the processing is too much for one processor.
Yep, it's fun to "tweak" the software to get what you want. Good going.

(03 Jun 2012) - Reply:
Hi Greg, I had issues with the two sets of timings going on, wheel counter & servo timing, so I opted for the simple solution. I'll detail what I did

Comment Greg Elmassian:
So you ran some tight loops and it actually did overload the processor, so to speak. Got it, they are cheap enough.

(04 Jun 2012) - Update:
I've just uploaded a video showing the latest software version hauling a heavier test train that I ran on Saturday morning.

Comment Eric Shade:
That really is pretty slick Dave, I am impressed! I love the sound of her working, very realistic.

Comment Pete Thornton:
Very impressive.

Comment Henner Meinhold:
Dave: you beat me with this project. I have been working on "slow motion" for many years, when I still built electric HO. See one of my achievements here:

My approach for steam would be to intercept the servo pulses, interpret them, compare with the real speed via slotted wheel and then generate new pulses for the throttle servo. I am confident this can be done with one tiny controller (I use the C8051F300 from Silabs - about 3mm by 3mm and assembly language). In the little Baldwin shown in the video I do DCC decoding, PID speed control with EMF feedback, smooth acceleration/deceleration, detection of missing rail contact and control of a servo (another project for a road/railbus) with this micro-controller. I have all the software and hardware bits and pieces for the steam project but alas have not found the time yet to implement it. So I am pretty curious how you get along with your speed control. One challenge would be very slow speed, as condensation will set in and torque will drop drastically.

Comment Bob Sorenson:
Hi Dave: Great addition to the deWinton!! Now you have me thinking. Would a mechanical governor in this scale be possible?

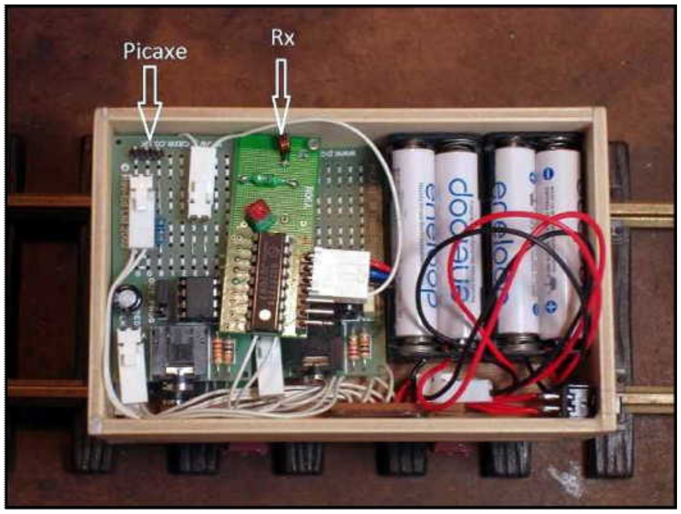
(04 Jun 2012) - Reply:
Henner: You are an expert! I have found that large throttle openings are more successful than smaller ones. And then quickly shutting off the throttle as soon as the speed has caught up.

(04 Jun 2012) - Reply:
Hi Bob: I think it would be possible, how about giving it a go?

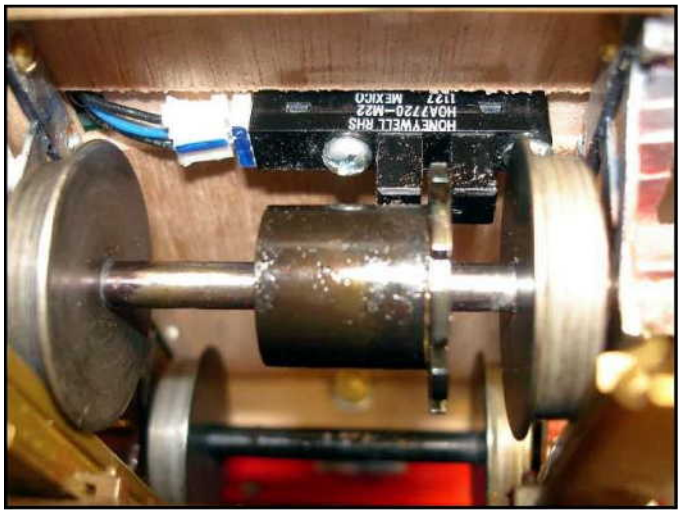
Comment Keith Yundt:
Interesting, but you're not the first. I built one to do the exact same thing around ten years ago using a Basic Micro Atom chip...works great. The trick is in the programming to get nice smooth and realistic control so it isn't jumpy.

Comment Keith Yundt:
Dave, Henner, one thing that really helped mine perform better was to program in some temporary "overshoot" of the throttle, with the exact amount and timing depending on the variance between actual speed and desired target speed.

(04 Jun 2012) - Update:
Here is the trailing car contents; all the control gear.



I'm using Picaxe M08M2 chips on three of their excellent AXE021 Proto Board Kits. These proto boards and chips are inexpensive. As per the Picaxe recommendations I'm using separate power supplies for the PICAXE's and for the servo power. These are 4 AAA cell Eneloop re-chargeables. The Picaxe cannot take 6 volts, so if non-rechargeables were used only a 3-cell battery pack should be used.

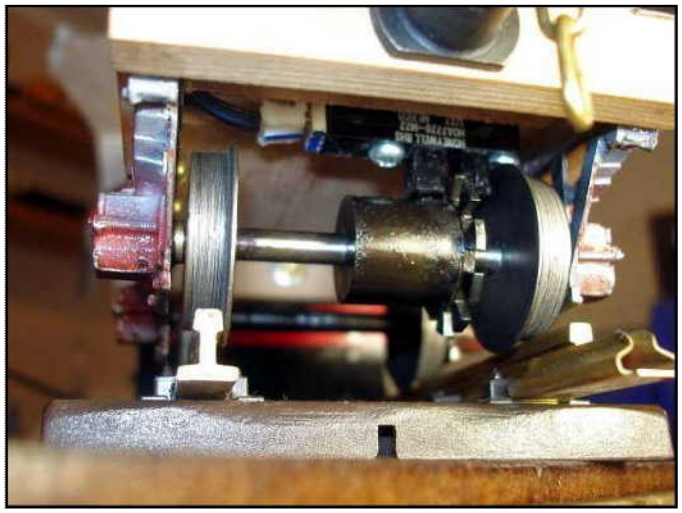


There's a lot of spaghetti wiring. All the PCBs, the IR sensor, the RC Rx, and the batteries are need to be well grounded to each other.

Here is the excellent Honeywell HOA7720-M22 IR sensor attached to the floor of the car.

This sensor is really good as it has an in-built Schmitt trigger; this provides a nice fat high/low output. It only needs three connections, the two power lines and a signal out.

Here is a shot of the slotted-disc type counter wheel attached to the axle.



I'm using a 10-slot counter wheel. This breaks the infrared beam from the sensor. The sensor converts this into the digital pulse. The PICAXE counts the pulses over a time frame, and then outputs whether the count is high or low.

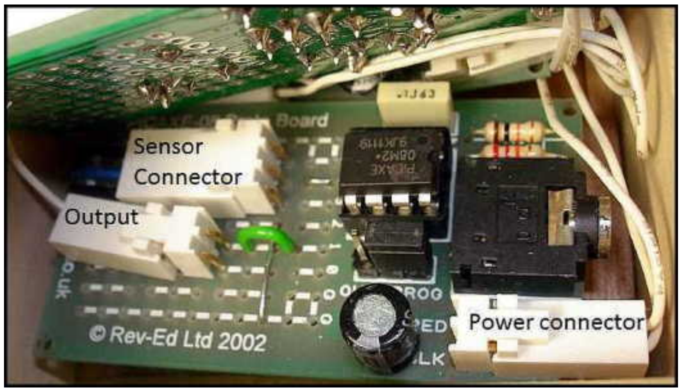
I'm using three Picaxe proto PCBs, the first counts the pulses, the second is the throttle servo controller, the third is the R/C receiver controller and whistle servo controller.

(04 Jun 2012) - Reply:
Hi Keith: thanks for your insight. It has been fascinating finding out about how the loco responds. I'm so pleased with it that I thought I'd describe what I did so that others can have a go if they would like.

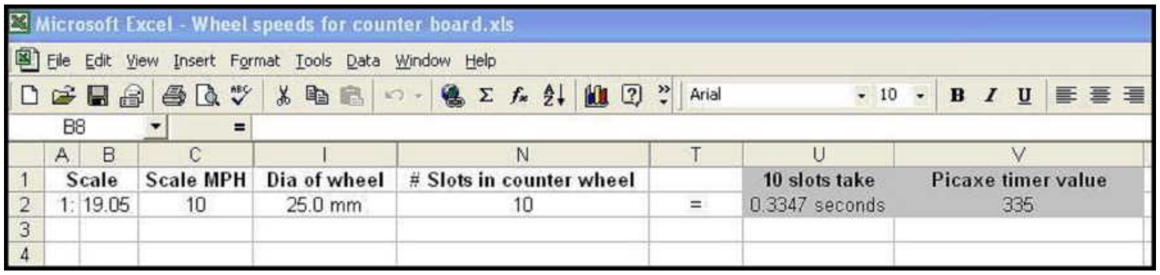
> ***Comment Keith Yundt:***
> *You did a great job Dave. I like the way you used the infrared sensor--that's an improvement on mine where I used a magnet wheel. I'm not sure how you are setting your target speed, but the way I did it was to use the A/D converter in the chip to interpret a potentiometer input, that way I could vary the speed on the fly. Once you get it to the speed you want, you have a button input to capture the #pulses at that speed and that becomes your new target.*

(04 Jun 2012) - Reply:
Hi Keith: My set-up has the counter board looking for 10 pulses from the sensor in a 1/3 second period, and if the count is high or low the output port goes high or low. I've just got the loco set for a scale 10 mph as I'm a narrow gauge man at the moment.

(05 Jun 2012) - Update:
Here's the counter board. The power connector feeds in the 5V from the batteries. The sensor connector supplies +5V and Gnd. from the board and receives the sensor signal from the central blue wire. The blue wire feeds through to port 4 on the Picaxe. The output from the Picaxe is on port 1. The output connects to the servo board.



Here's a shot of my Excel spreadsheet to calculate the timing.



Microsoft Excel - Wheel speeds for counter board.xls

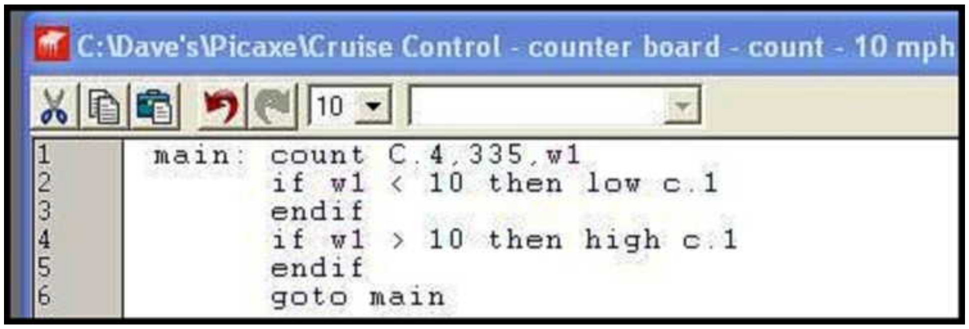
	A	B	C	I	N	T	U	V
1	Scale		Scale MPH	Dia of wheel	# Slots in counter wheel		10 slots take	Picaxe timer value
2	1:	19.05	10	25.0 mm	10	=	0.3347 seconds	335
3								
4								

The spreadsheet was used to calculate the Picaxe timer value for 10 slots of the counter wheel at a scale 10 mph.

I can email you a copy of my spreadsheet if you wish, please email me:

bouser@gmail.com

Here's a shot of the Picaxe programmer showing the simple instructions that I am using.

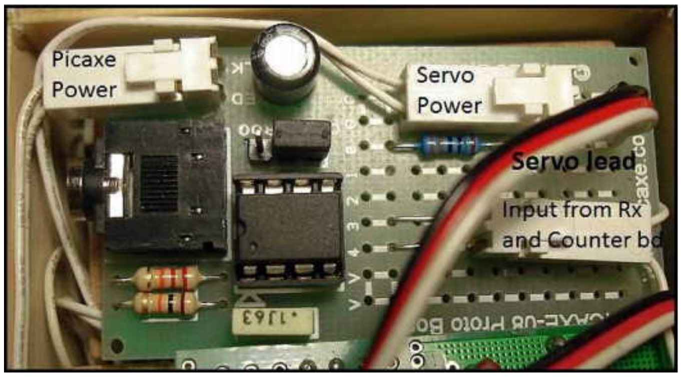


C:\Dave's\Picaxe\Cruise Control - counter board - count - 10 mph

```
main:  count C.4,335,w1
       if w1 < 10 then low c.1
       endif
       if w1 > 10 then high c.1
       endif
       goto main
```

The instruction states to count the pulses on port 4 for 0.335 seconds, and store the number as w1. Then if the number is less or more than 10 set the output port 1 high or low.

Picaxe starter kits AXE003/AXE003U contain the proto board, the chip, the programming cable, and all the software required.



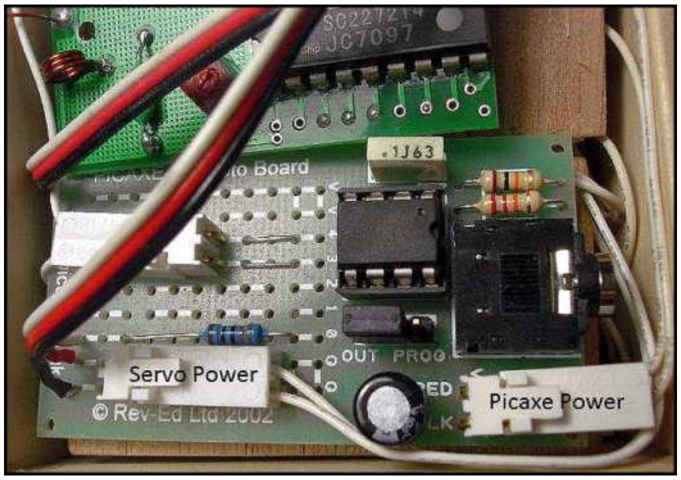
(06 Jun 2012) - Update:
Here is the throttle servo board.

This board has a separate power supply for the servo, this was recommended to keep "noise" out of the Picaxe power supply. You can see the servo connect cable; it is now soldered direct to the board. The colour code of the cable is red for +5V, black Gnd; white is the signal from the PICAXE. There is a 330-Ohm resistor between the Picaxe output on port 0 and the white servo lead.

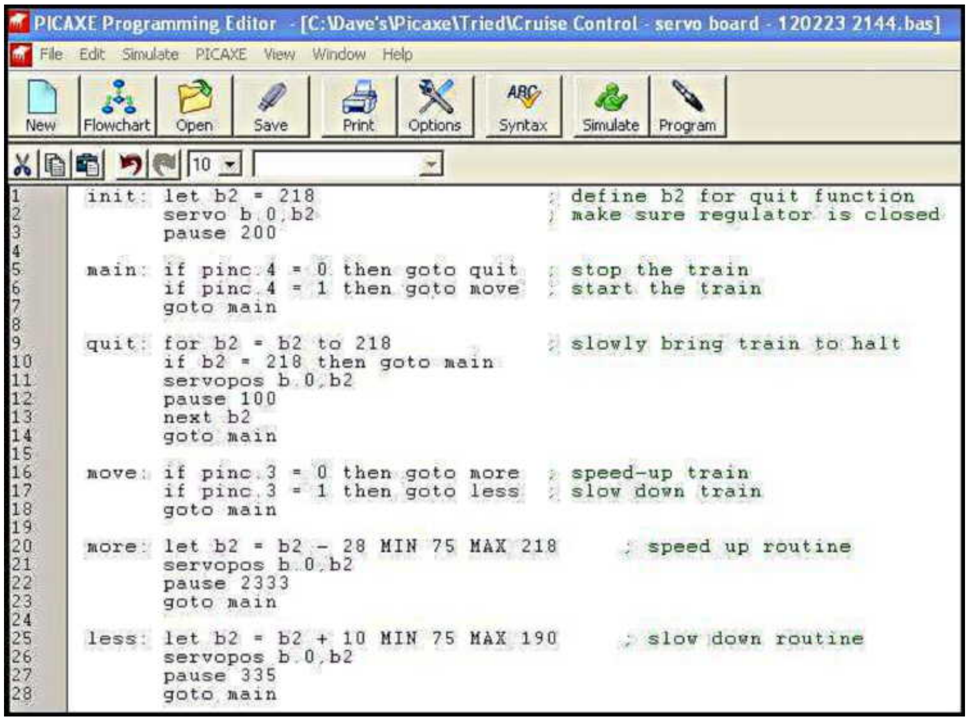
The Gnd for both power supplies are connected on the boards - the lower track on the image below is the 0V track. It is essential that all the boards, the Rx, and the IR sensor all share a common ground.

The other white connector has the signal from the receiver - Rx connected direct to Picaxe input port 4, and the signal from the counter board connected to Picaxe input port 3.

Here is another view of the throttle servo board with the Rx above it.



Here is a screenshot of a Picaxe program that I have used.



PICAXE Programming Editor - [C:\Dave's\Picaxe\Tried\Cruise Control - servo board - 120223 2144.bas]
File Edit Simulate PICAXE View Window Help
New Flowchart Open Save Print Options Syntax Simulate Program

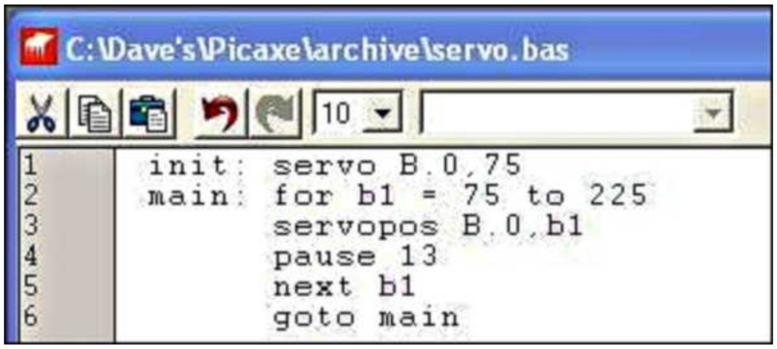
```
init:   let b2 = 218                        ; define b2 for quit function
        servo b.0,b2                        ; make sure regulator is closed
        pause 200

main:   if pinc.4 = 0 then goto quit        ; stop the train
        if pinc.4 = 1 then goto move        ; start the train
        goto main

quit:   for b2 = b2 to 218                  ; slowly bring train to halt
        if b2 = 218 then goto main
        servopos b.0,b2
        pause 100
        next b2
        goto main

move:   if pinc.3 = 0 then goto more        ; speed-up train
        if pinc.3 = 1 then goto less        ; slow down train
        goto main

more:   let b2 = b2 - 28 MIN 75 MAX 218         ; speed up routine
        servopos b.0,b2
        pause 2333
        goto main

less:   let b2 = b2 + 10 MIN 75 MAX 190         ; slow down routine
        servopos b.0,b2
        pause 335
        goto main
```

This program at switch-on moves the throttle to the stop position if it isn't already. Then the program waits for the start button press on the transmitter. When the button is pressed the program shoots down to the more: section as the loco is stationary and applies the throttle, then waits for the loco to build-up speed, then returns to the main: section. In the main: section the program checks to see if the stop button has been pressed. Then the program proceeds to the move: section to check whether the loco is above or below speed. And so on.

I'm using a more complicated version now but this one explains the system easily.
Next I'll deal with the throttle servo positions.

(07 Jun 2012) - Update:
Throttle Servo Positions: The throttle servo needs to be mounted to the loco and hooked-up to the throttle just like on a radio control set-up. I'm using a Futaba mini servo on mine.

The next thing is to determine the required servo positions. This little program can be loaded on to the Picaxe, it is using port 0 as the servo output. This program sets the servo swinging over its entire range of movement (position 75 to position 225). The program can be altered to hold the servo in any of the positions.



```
init:  servo B.0,75
main:  for b1 = 75 to 225
       servopos B.0,b1
       pause 13
       next b1
       goto main
```

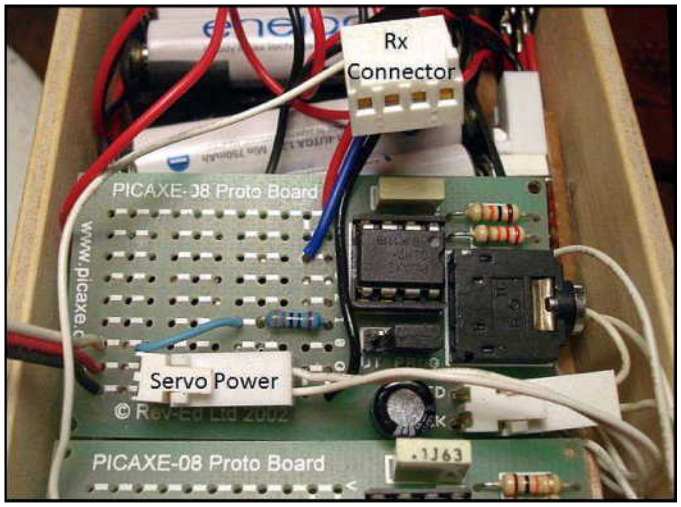
My set-up now has throttle fully closed at pos. 225 and my largest throttle opening is set to 80. The secret to my set-up is the pre-set minimum throttle position for when the loco is running. For me this is the required throttle setting for my loco to cruise along my straight level sections. These are my current throttle settings for running; they feature in my video of Albus hauling the heavier test train:

225 -	Fully shut off (quit: position)
201 -	Minimum setting for running
161 -	1st application of throttle after starting sequence
147 -	Starting throttle position / 2nd application of throttle
100 -	3rd application of throttle
80 -	4th application of throttle

Comment John *(MLS - Ora Banda)***:**
Intriguing stuff, Dave... thanks for taking the time to share this in such detail. I might have to add this to my own "to-do-list". Where did you get the Honeywell sensor from?

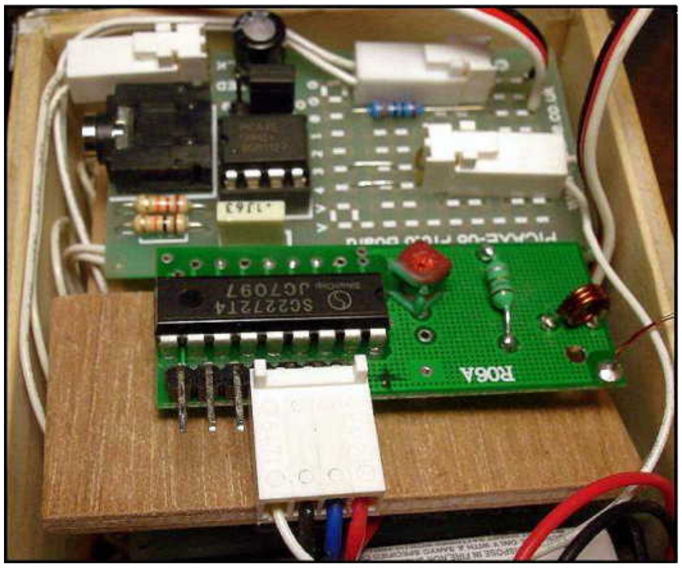
(07 Jun 2012) - Reply:
Hi John: I bought it online from **Element 14** (formerly Farnell), I think it came from their Aus branch. And RS do them online as well.

(08 Jun 2012) - Update:
It's time for the whistle servo board and the receiver.

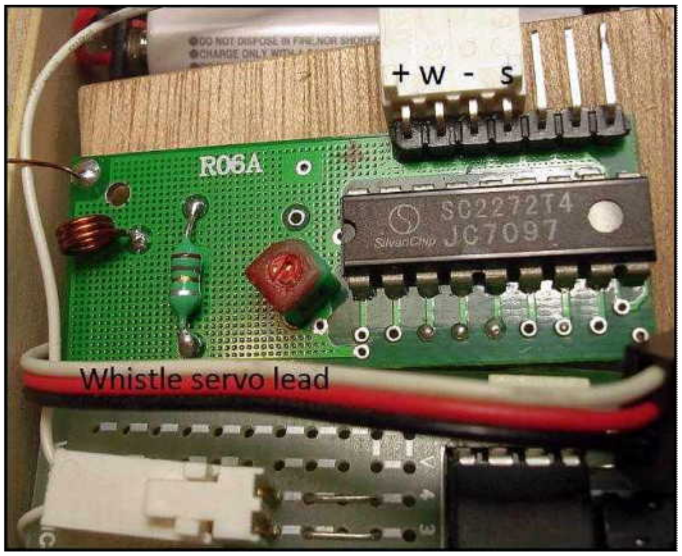


Here is the whistle servo board. The receiver connector is also visible.

The whistle servo board provides power for the receiver, and takes the whistle signal from the Rx on input port 4.

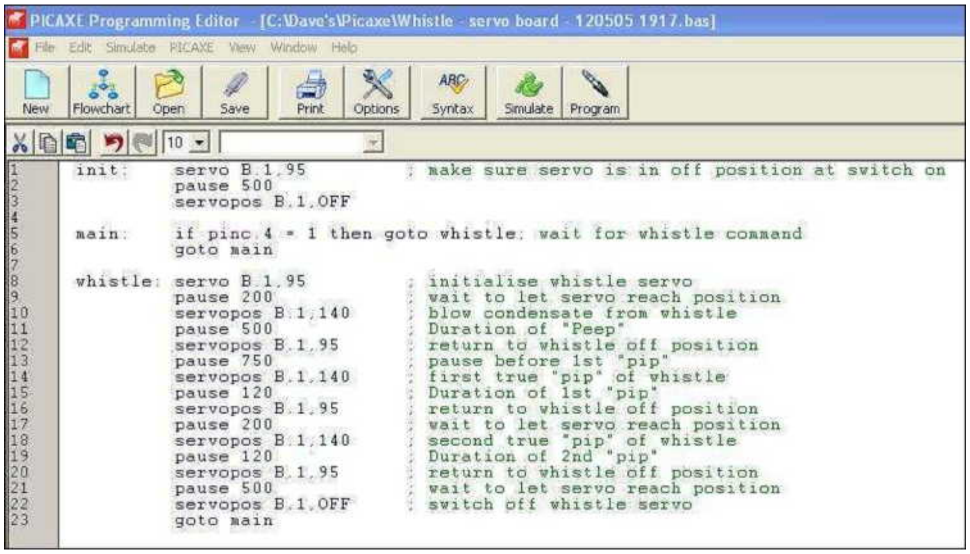


This image shows the Rx and the regulator servo board. The whistle board is under Rx.



Here is the receiver with the connector in place. There are three channels that aren't being used.

Here's the program for the current "peep-pip-pip" whistle sequence.


PICAXE Programming Editor - [C:\Dave's\Picaxe\Whistle - servo board - 120505 1917.bas]
File Edit Simulate PICAXE View Window Help
New Flowchart Open Save Print Options Syntax Simulate Program
10

```
init:     servo B.1,95          ; make sure servo is in off position at switch on
          pause 500
          servopos B.1,OFF

main:     if pinc.4 = 1 then goto whistle; wait for whistle command
          goto main

whistle:  servo B.1,95          ; initialise whistle servo
          pause 200             ; wait to let servo reach position
          servopos B.1,140      ; blow condensate from whistle
          pause 500             ; Duration of "Peep"
          servopos B.1,95       ; return to whistle off position
          pause 750             ; pause before 1st "pip"
          servopos B.1,140      ; first true "pip" of whistle
          pause 120             ; Duration of 1st "pip"
          servopos B.1,95       ; return to whistle off position
          pause 200             ; wait to let servo reach position
          servopos B.1,140      ; second true "pip" of whistle
          pause 120             ; Duration of 2nd "pip"
          servopos B.1,95       ; return to whistle off position
          pause 500             ; wait to let servo reach position
          servopos B.1,OFF      ; switch off whistle servo
          goto main
```

This program switches off the whistle servo when not in use, and hopefully saves plenty of power. There are pauses in the sequence to allow the servo to return to the whistle off position (position 95). Servos aren't too fast on 5V. I'm using a cheap micro servo for the whistle.
I think this installment wraps up the project.

Best wishes,
Dave.