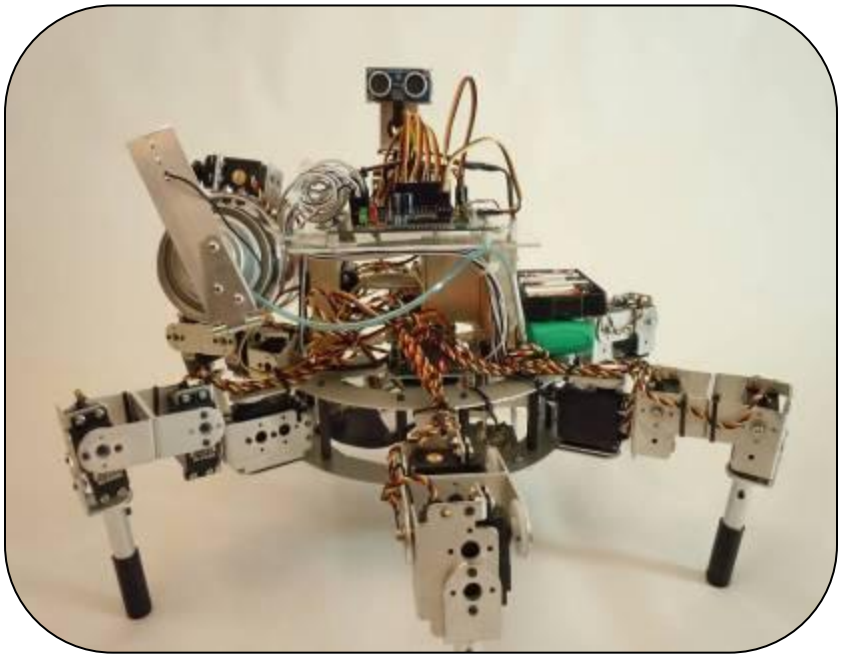
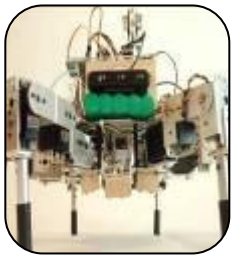




Prospero: Robotic Farmer ©

Single Member of A Robotic Swarm

David Dorhout

vanmunch@yahoo.com

YouTube: vanmunch36

December 31st, 2010

SchmartBoard Propeller Design Contest

Project Number: PP007

Table of Contents

Project Number

Project Number: PP007

Project Description

Robotics in Agriculture

The past hundred years have seen the greatest changes as the power of science transforms our world including farming. Despite its quaint reputation, agriculture has always been an early adapter of technology. This is evident from the beginning of mechanization with the cotton gin, McCormick's Reaper, tractors, hybrid seed, to genetically engineered plants that protect themselves and grow in arid environments. Yields have grown at an amazing pace, but demand from developing countries and population growth exceed all of our best efforts.

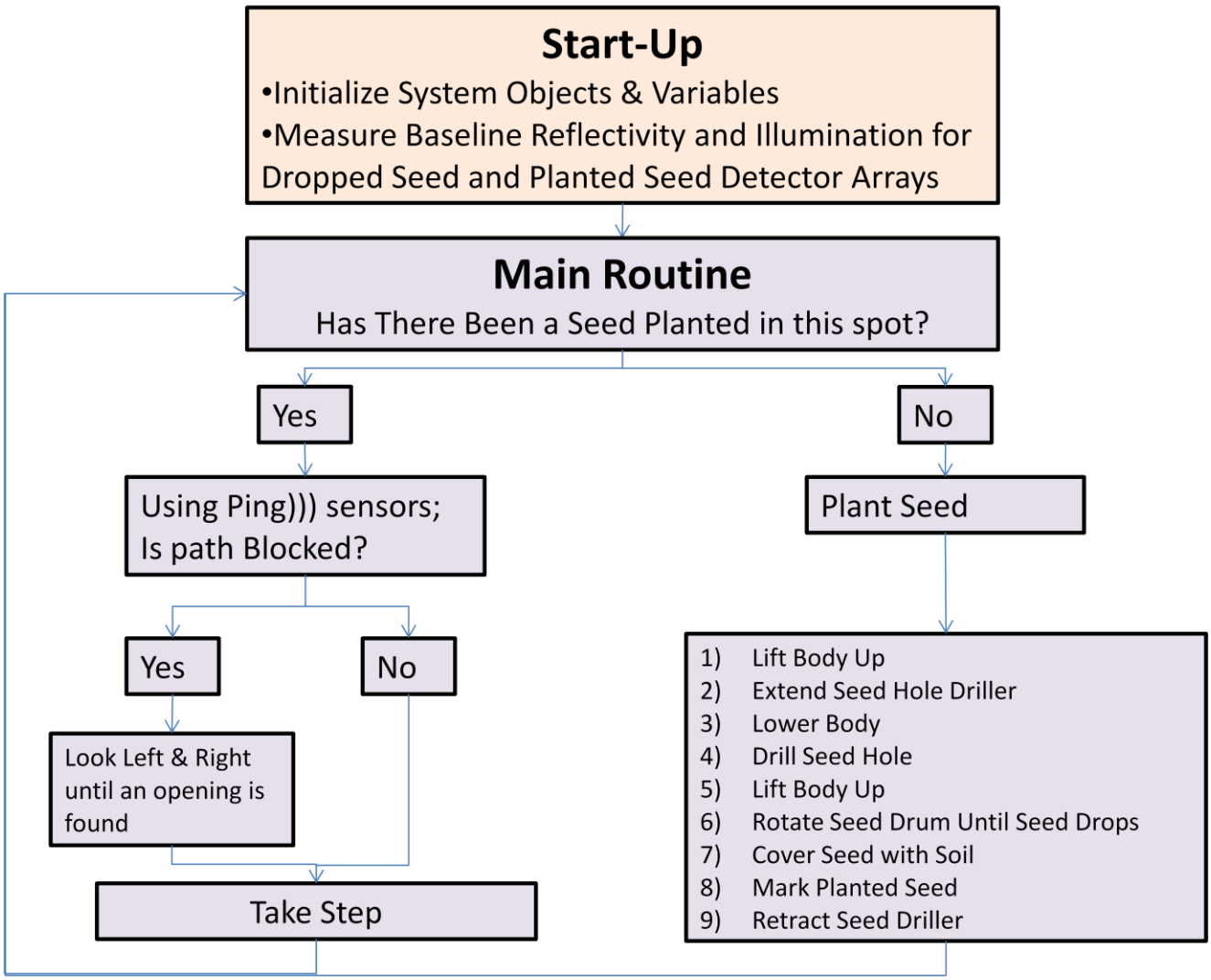
We know that we need to continue to find ways to increase the productivity of land on a per unit basis. To this end, agriculture has started to add computerization and automation to the existing machinery with things like aftermarket GPS farming systems that can autonomously drive tractors, monitor yield, and apply fertilizer. However, these aftermarket add-ons are not the well thought-out, sophisticated systems that you would see in an industrial manufacturing facility. When those companies would build a new factory utilizing robotics, they wouldn't just stick a robot in the place where a person stood. They would completely reexamine the process. They knew that they could break a complicated process into little parts to best fit the application. These little processes can be controlled by multiple, inexpensive controllers working independently while making independent decisions, but with the same goal.

Today's agricultural equipment has been designed around the controller, a person sitting in a chair. It cost a lot to employ this single operator so the equipment grew larger in order to maximize the productivity of that one person. However, this method has its drawbacks. Nature is chaotic and dynamic. Soil nutrients and moisture change from foot to foot. Having equipment that allows a single person to plant a thousand acres in a day comes at the cost of productivity per acre as a result of having to treat all the acres as the same. So, do we have to sacrifice productivity per acre for productivity per person?

Prospero is the working prototype of an Autonomous Micro Planter (AMP) that uses a combination of swarm and game theory and is the first of four steps. It is meant to be deployed as a group or "swarm". The other three steps involve autonomous robots that tend the crops, harvest them, and finally one robot that can plant, tend, and harvest--autonomously transitioning from one phase to another.

Prospero is controlled with a Parallax Propeller chip. The powerful, eight independent processors (cogs) allow for true parallel processing. The propeller chip is mounted on a Schmart Board allowing for access to all of the pins for rapid prototyping. Its hexapod body can autonomously walk in any direction, avoiding objects with its duel ultrasonic Ping))). Its walking algorithms allow it to instantly change direction and walk in any new direction without turning its body. An underbody sensory array allows the robot to know if a seed has been planted in the area at the optimal spacing and depth. Prospero can then dig a hole, plant a seed in the hole, cover the seed with soil, and apply any pre-emergence fertilizers and/or herbicides along with the marking agent. Prospero can then signal to other robots in the immediate proximity that it needs help planting in that area or that this area has been planted and to move on via coded IR transmissions that are currently represented with a green and red LED so that people can see it working. The more seeds it plants, the more the "green" LED lights up, the more it draws other robots nearby (+2). The more it detects planted seeds, the more it repulses other robots with the "red" LED (-1)

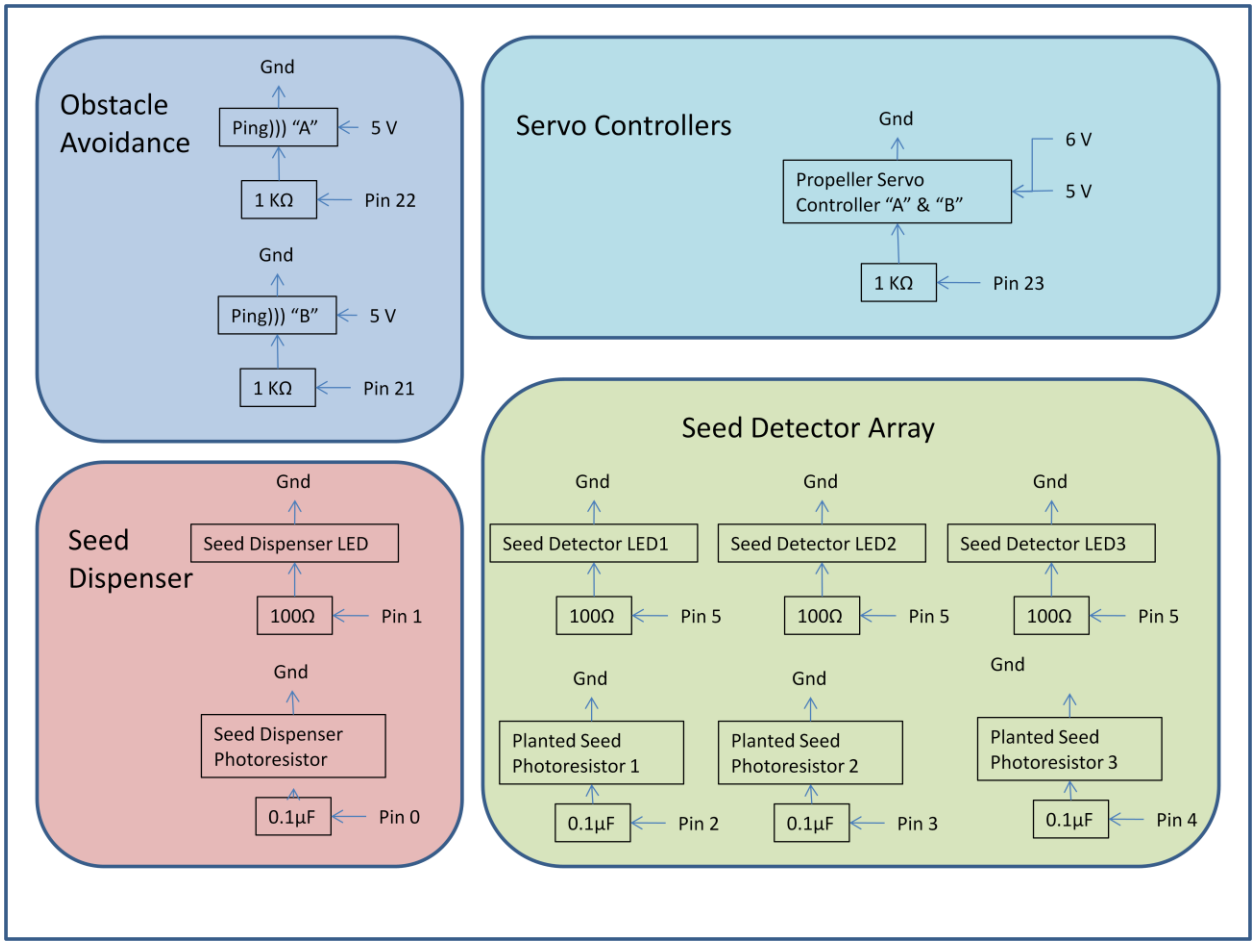
Block Diagram



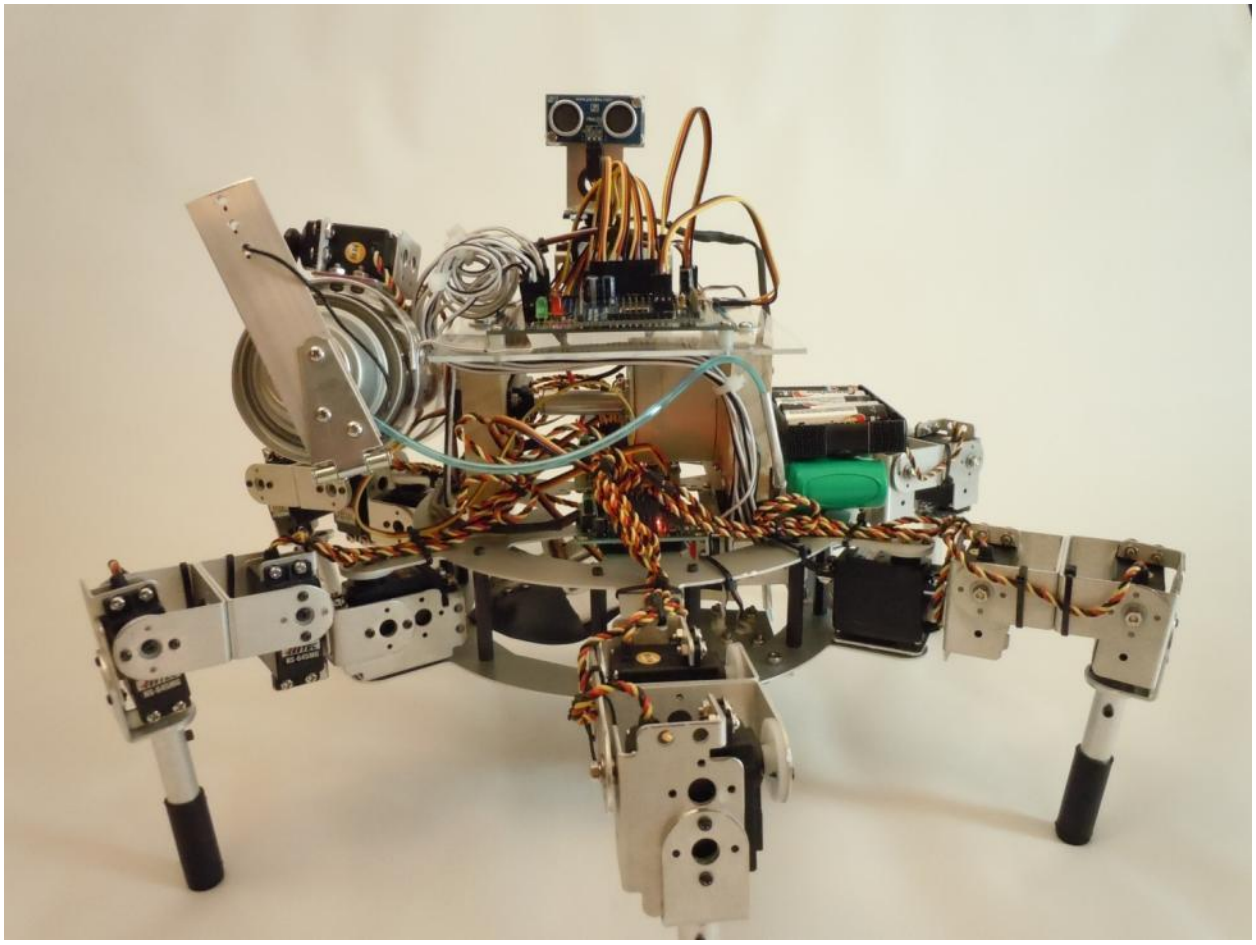
Bill Of Materials

Qty	Description	Company	Part Number
1	AH3-R (no electronics; no servos)	Lynxmotion	AH3RCA
1	6.0 V Ni-MH 2800 mAh Battery	Lynxmotion	BAT-05
21	HS-645 Servo	Tower Hobby	LM3122
1	HSR-1425CR Servo	Lynxmotion	S1425CR
8	Extender Cable- 6"	Lynxmotion	SEA-01
2	Aluminum Tubing - 3.0"	Lynxmotion	AT-02
1	Servo Horn Tubing Adapter	Lynxmotion	HUB-14
1	Aluminum Tubing Connector Hub	Lynxmotion	HUB-08
2	Aluminum Multi-Purpose Servo Bracket	Lynxmotion	ASB-04
1	Parallax Propeller SchmartModule	Schmart Board	710-0005-01
2	Propeller Servo Controller USB	Parallax	28830
1	PING))) Ultrasonic Sensor with Mounting Bracket	Parallax	910-28015A
1	PING))) Ultrasonic Sensor	Parallax	28015
1	Parallax Blank 3x4 Proto Board	Parallax	45305
4	0.1 uF Mono Radial Capacitor	Parallax	200-01040
4	Photoresistor - VT935G-B	Parallax	350-00009
4	100 ohm Resistor, 1/4 Watt	Parallax	150-01011
3	10mm Ultra-High Brightness Blue LED	Radio Shack	276-006
1	5mm High-Brightness White LED	Radio Shack	276-017
1	22awg, Solid, Black	Jameco Electronics	36792
1	22awg, Solid, Red	Jameco Electronics	36856
7	Unshrouded Header 3 Position 2.54mm Solder Straight Thru-Hole	Jameco Electronics	421489
7	Connector Housing 3 Position 2.54mm Straight	Jameco Electronics	157383
15	Connector Contact PIN 1 Position Crimp Straight Cable Mount Reel	Jameco Electronics	100766
1	1/8" OD; 0.066" ID Transparent Blue Polurethane tubing	ProTubing.com (Freelin-Wade)	1J-013-27
1	1/2" ID; Oilite Sintered Bronze Flanged Sleeve Bearing	Small Parts	B000FMUB66
1	White Spray Paint	Various	-
	Aluminum Tubing, Sheeting and Rods	Various	-
	1/2" Wood Board	Various	-
	1/8" Plexiglas	Various	-
	2" PVC Pipe	Various	-

Schematic

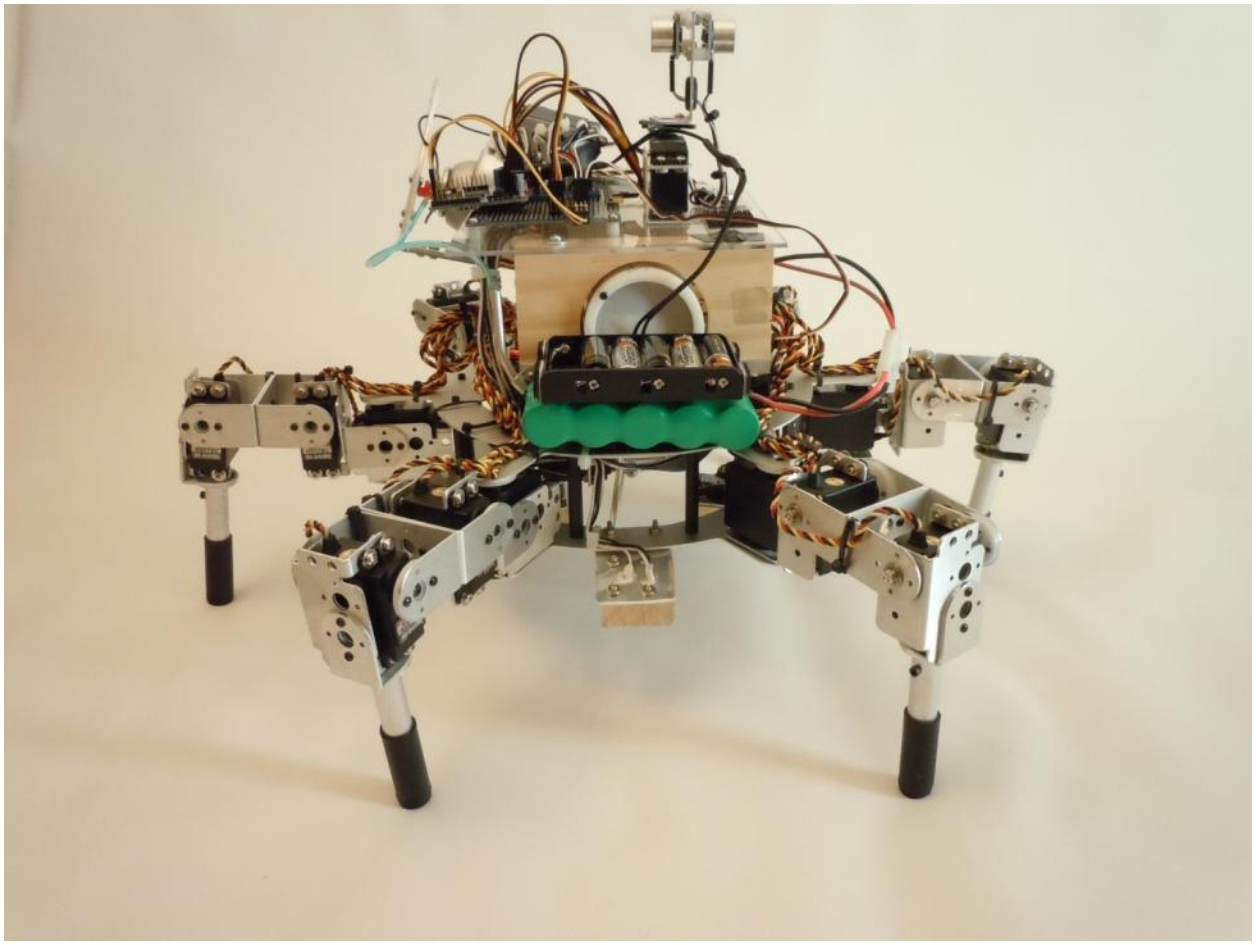


Photographs



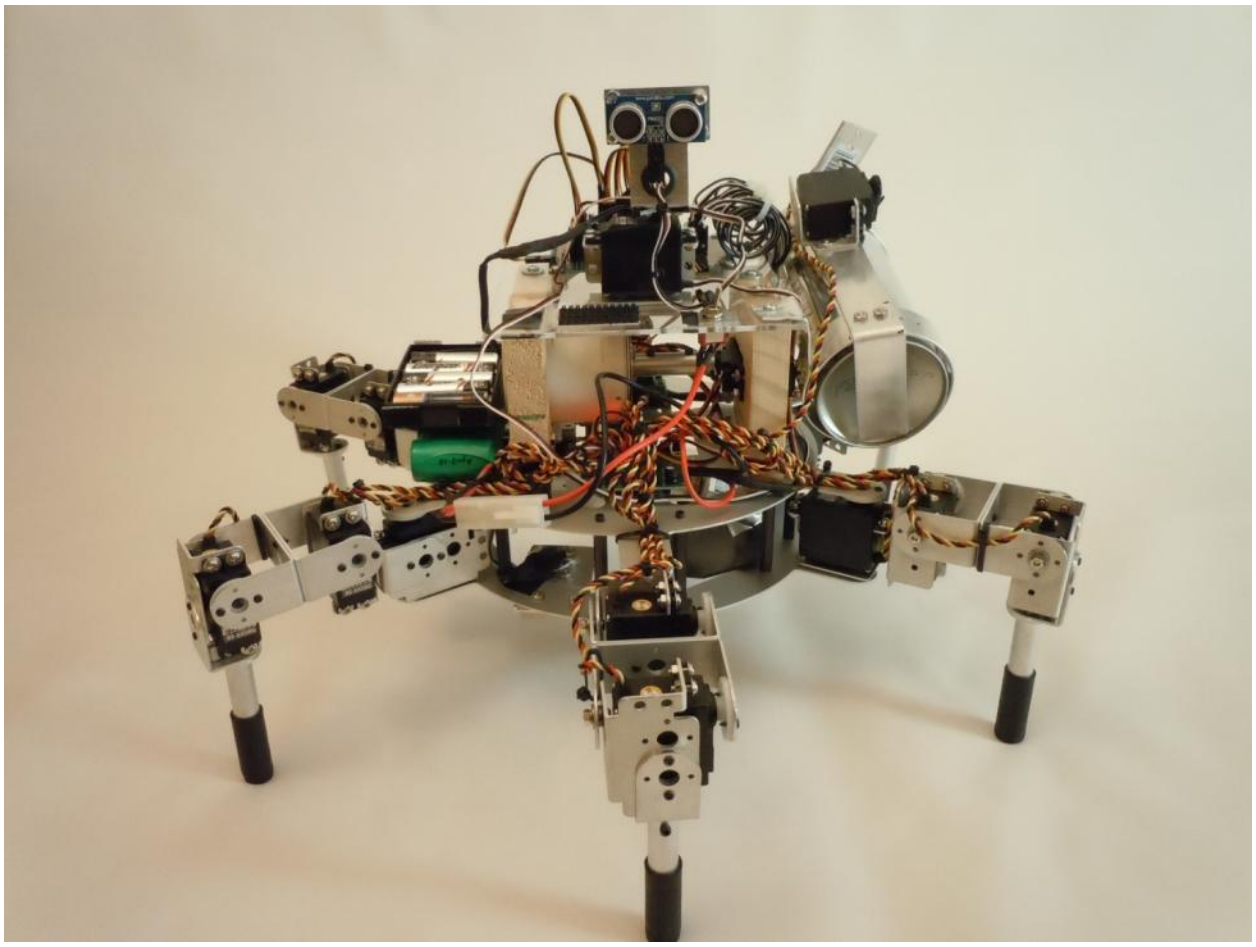
"Front*" view

*Prospero is functionally symmetrical and does not have true "sides"



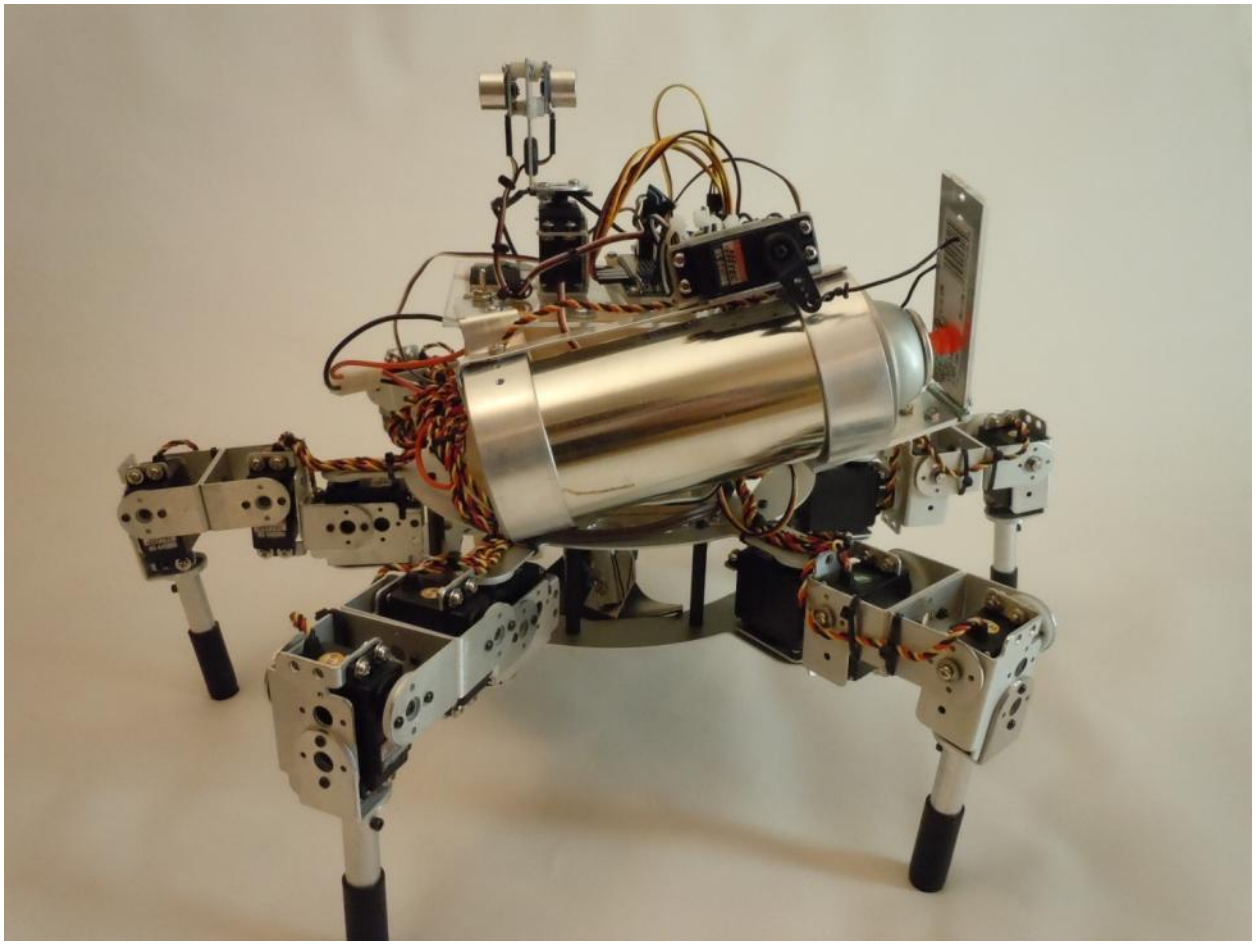
"Left*" view

*Prospero is functionally symmetrical and does not have true "sides"



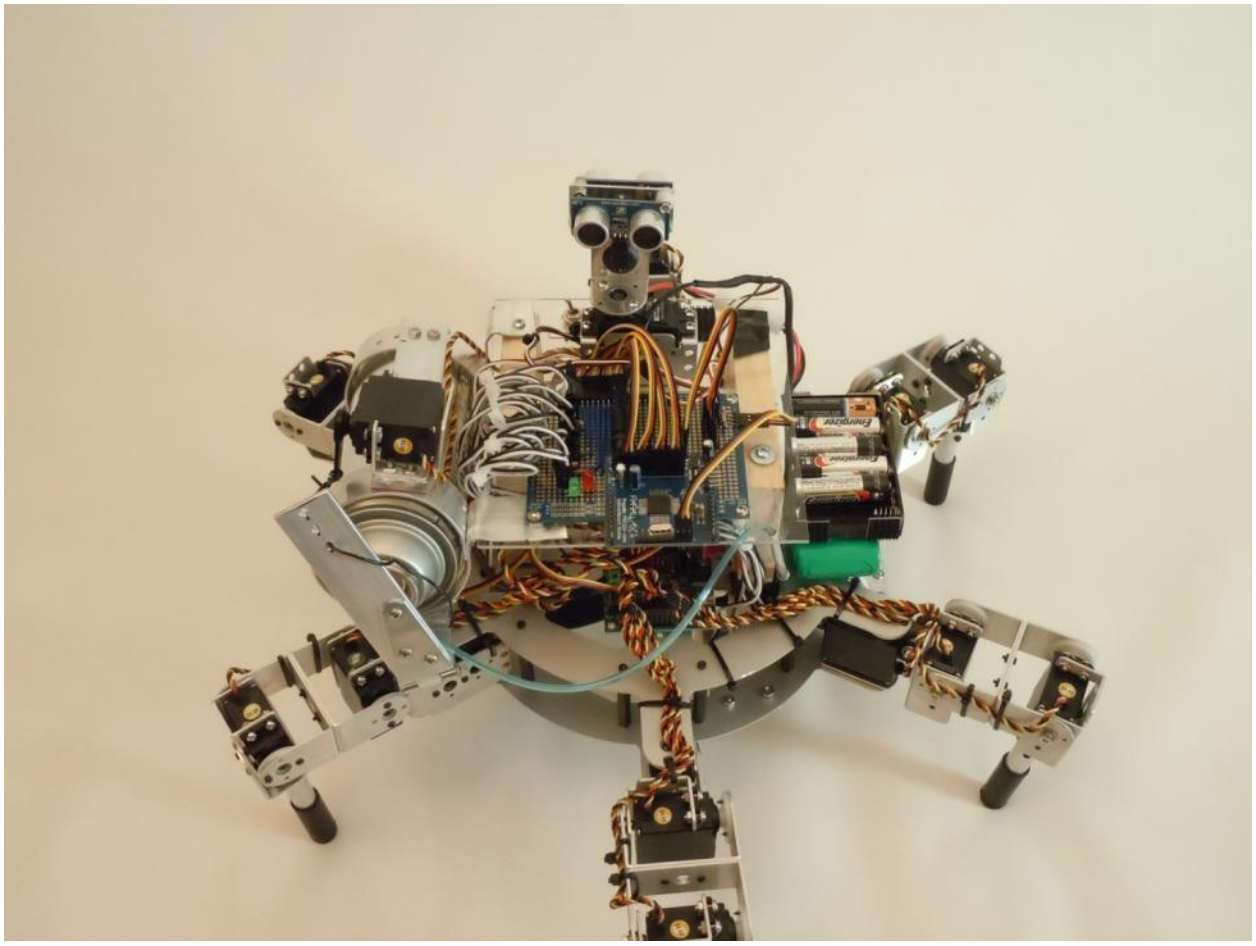
"Rear*" view

*Prospero is functionally symmetrical and does not have true "sides"

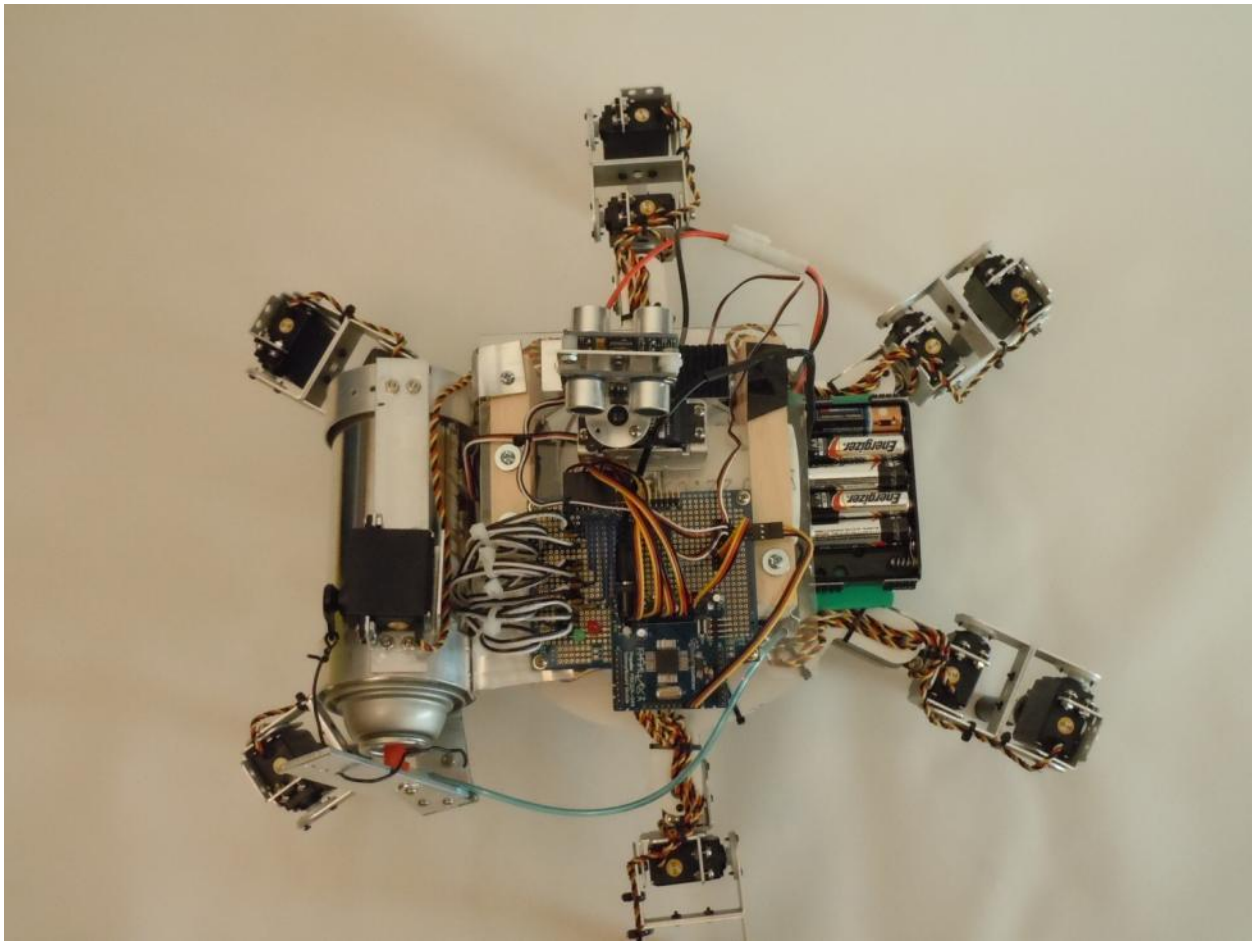


"Right*" view

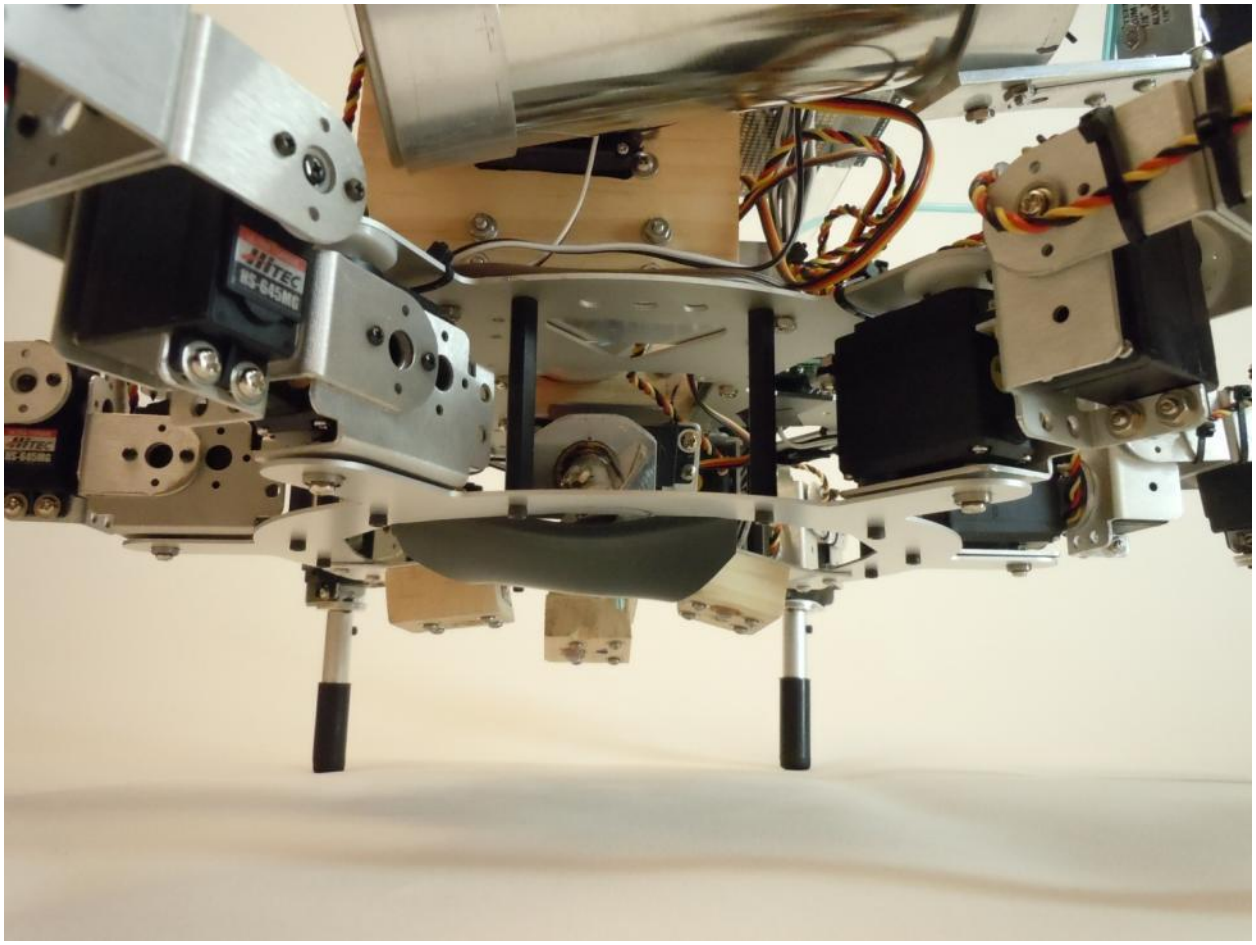
*Prospero is functionally symmetrical and does not have true "sides"



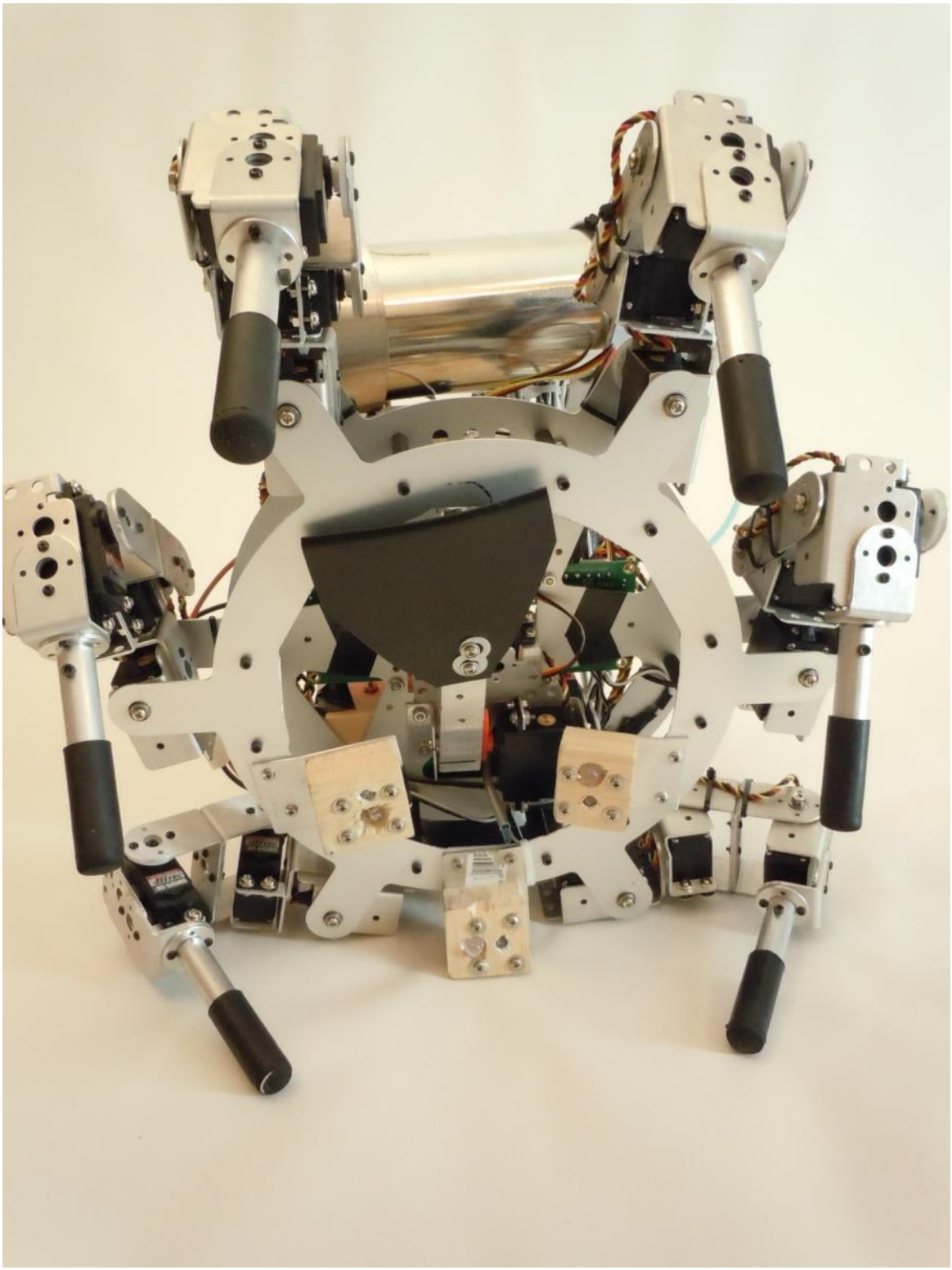
top view



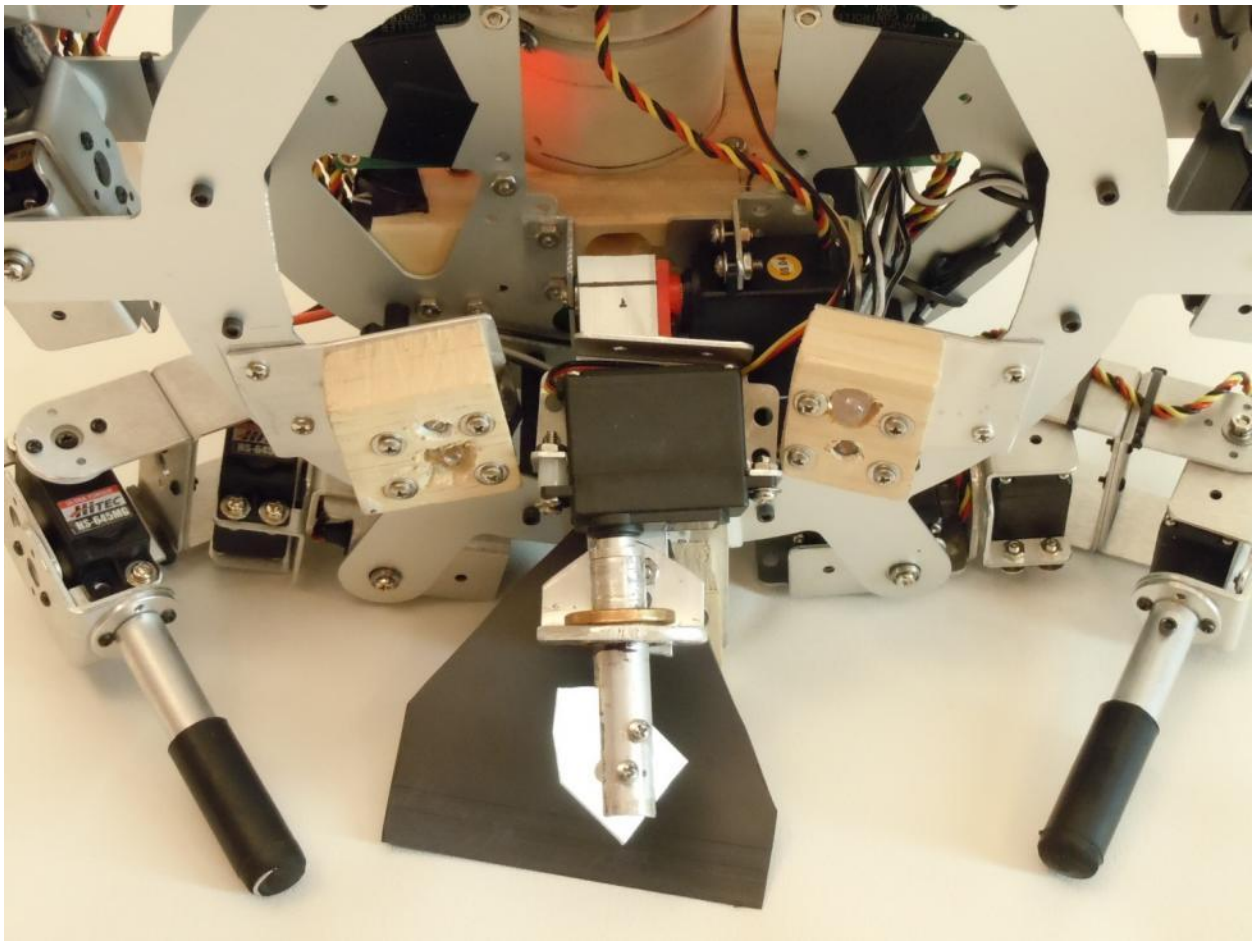
Overhead view



Under View

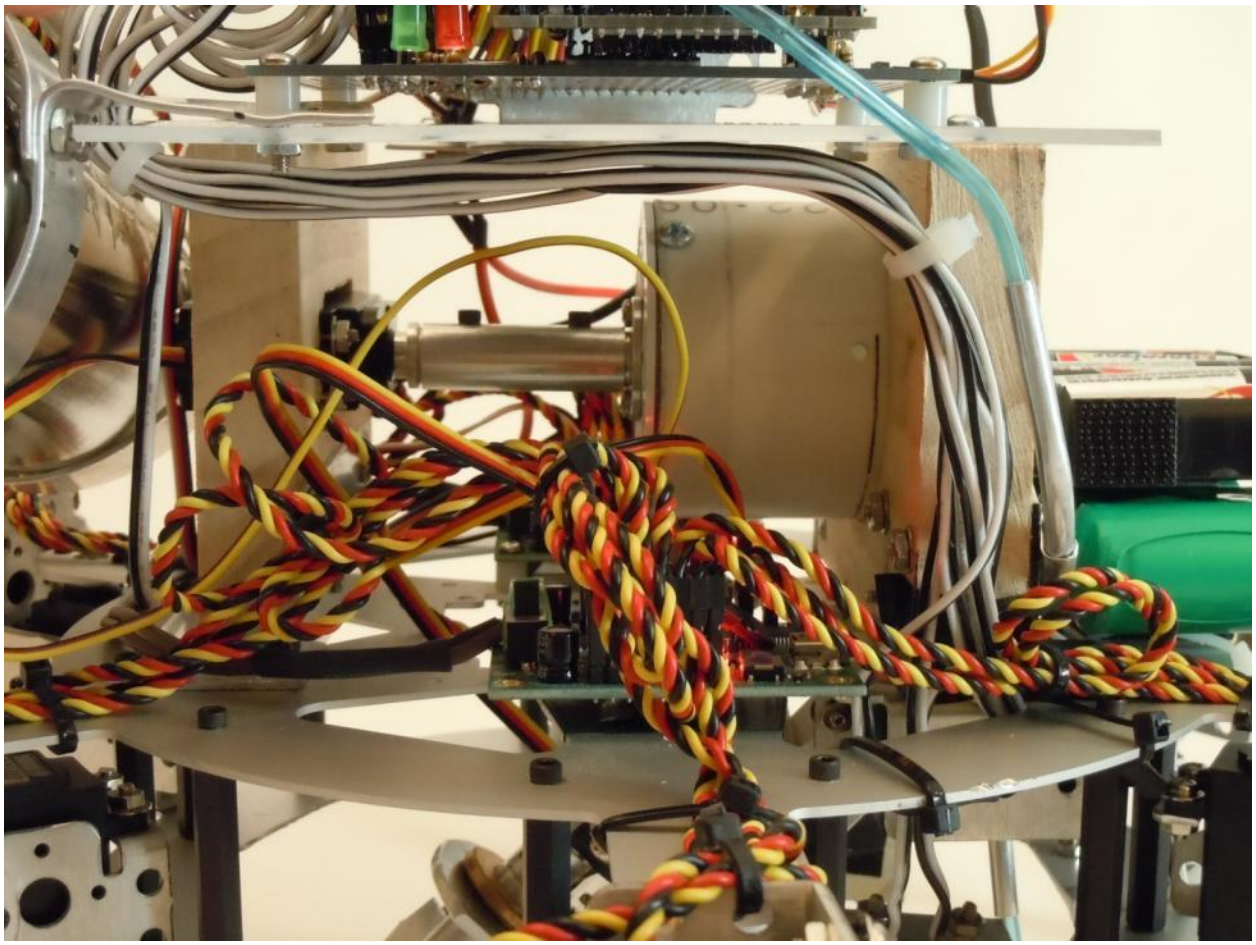


Under View showing Seed Detector Array and Seed Driller Retracted

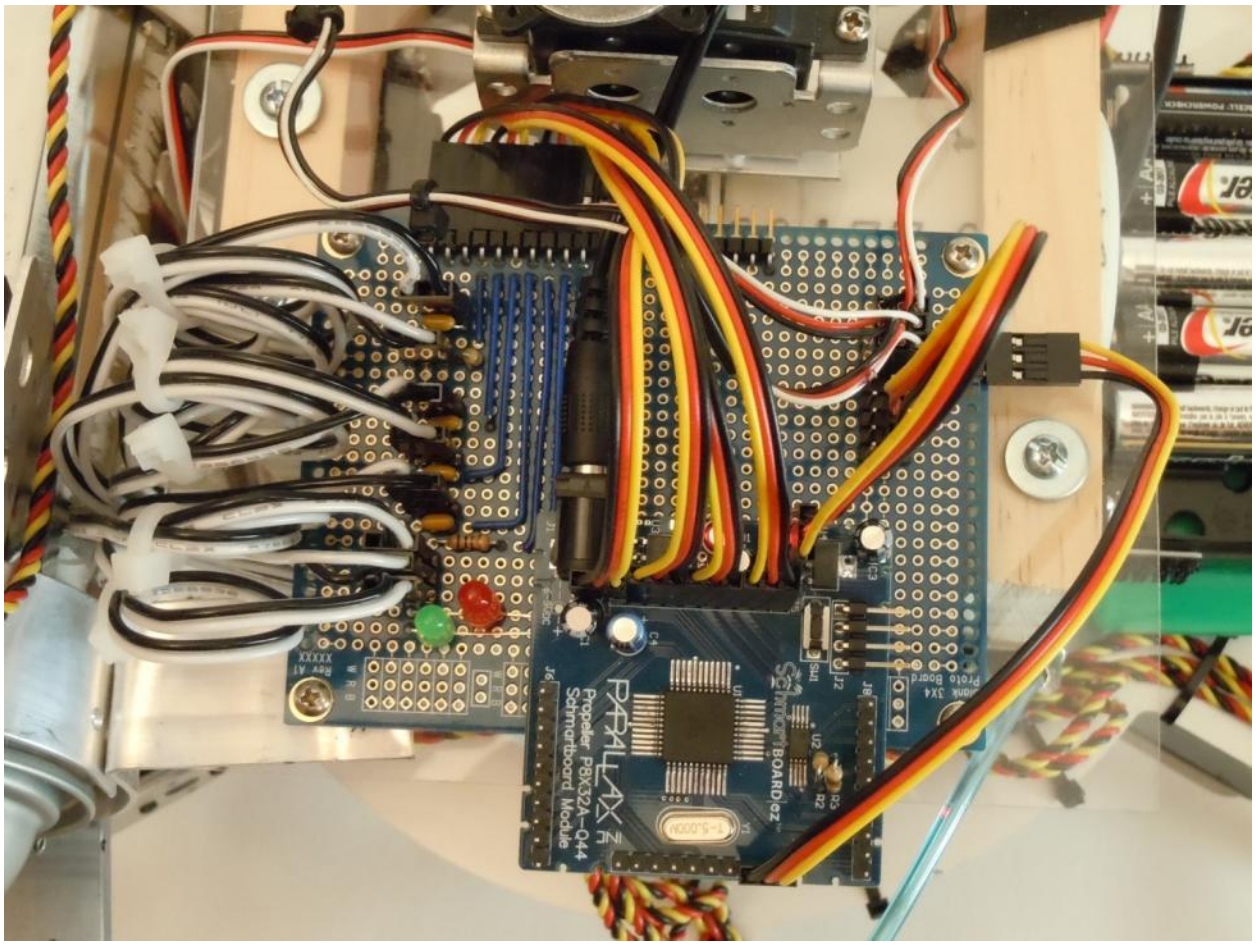


Under View showing Seed Detector Array and Seed Driller Extended

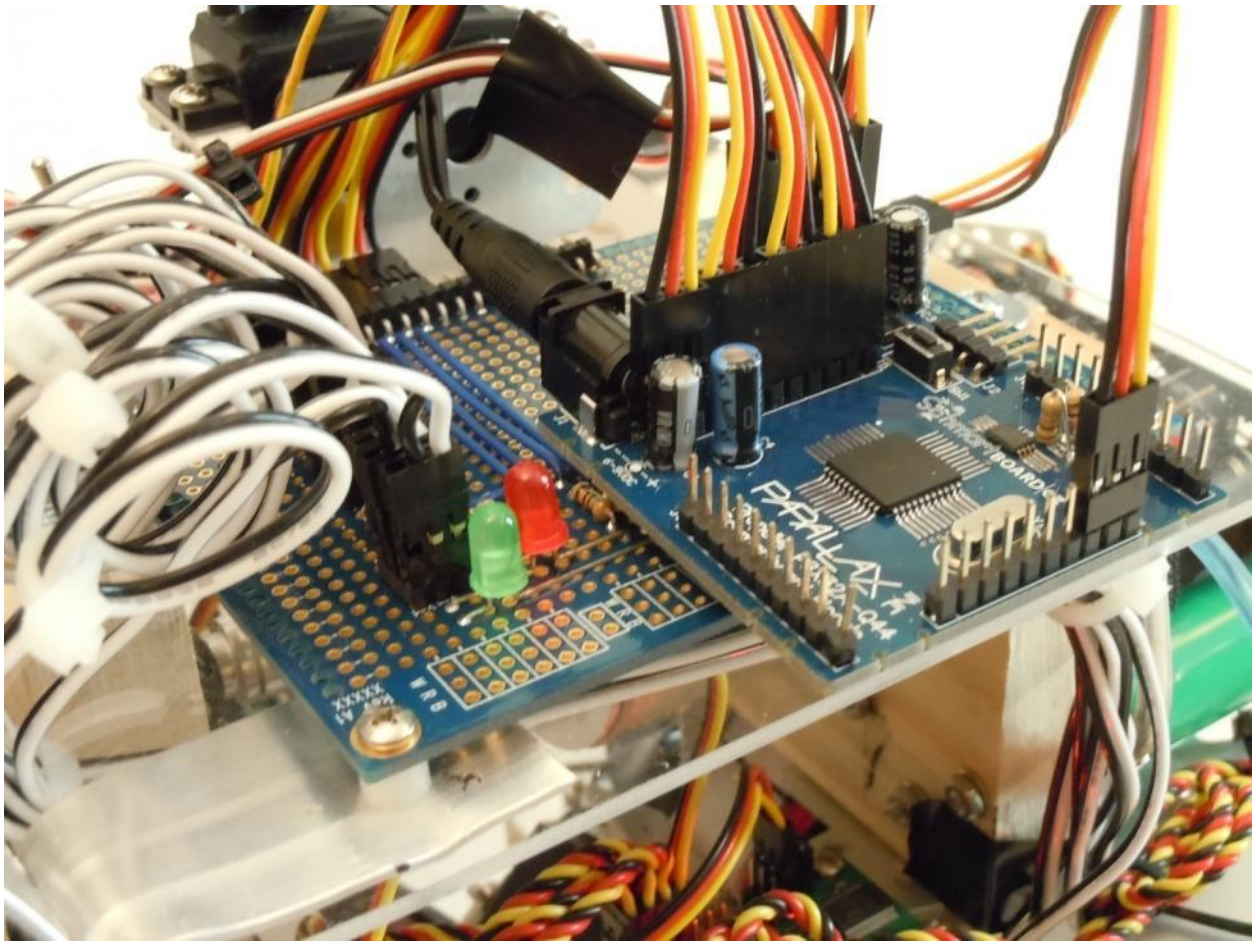
Seed drops through hole in wood, through square tube, and into drilled hole



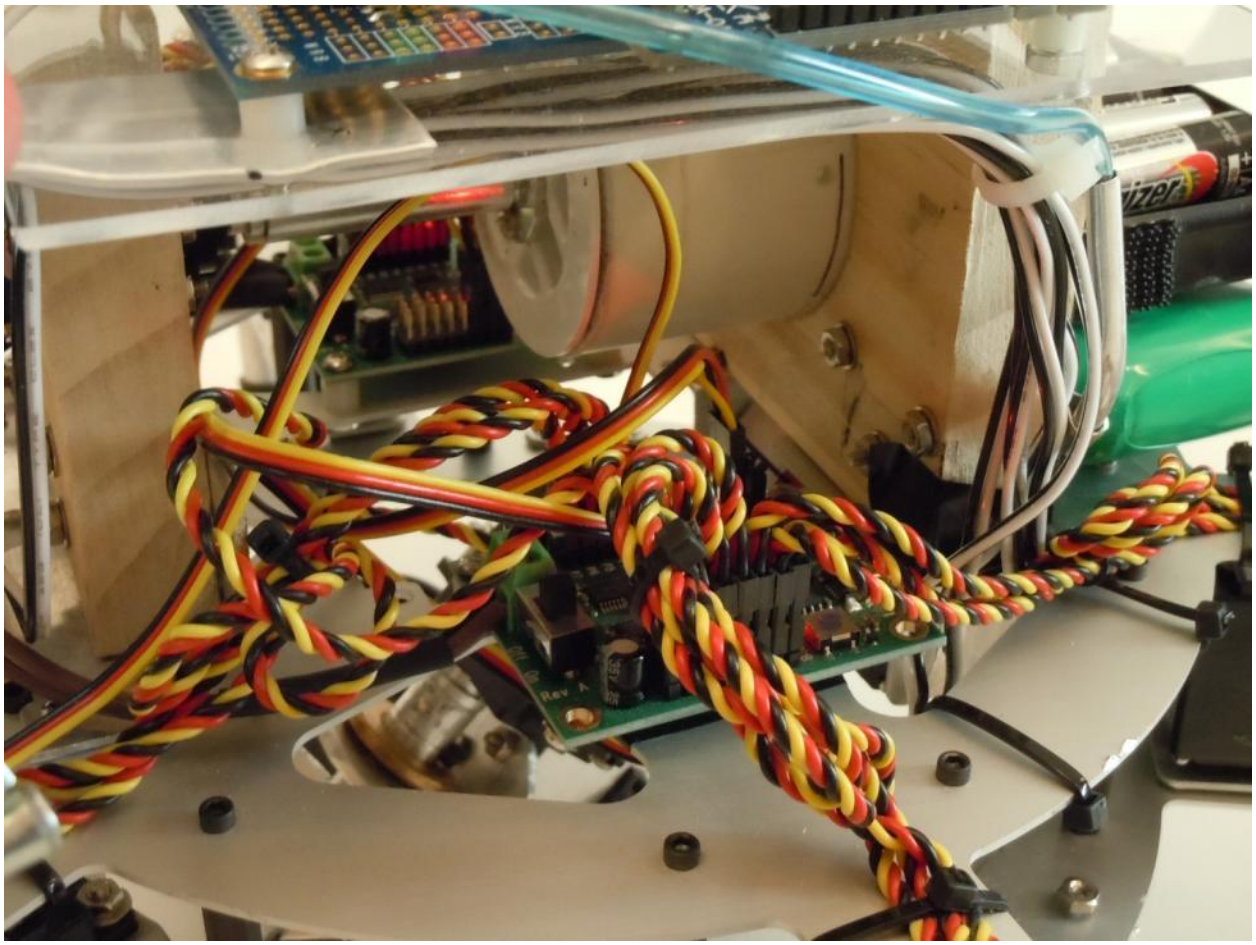
Side View Showing Seed Drum



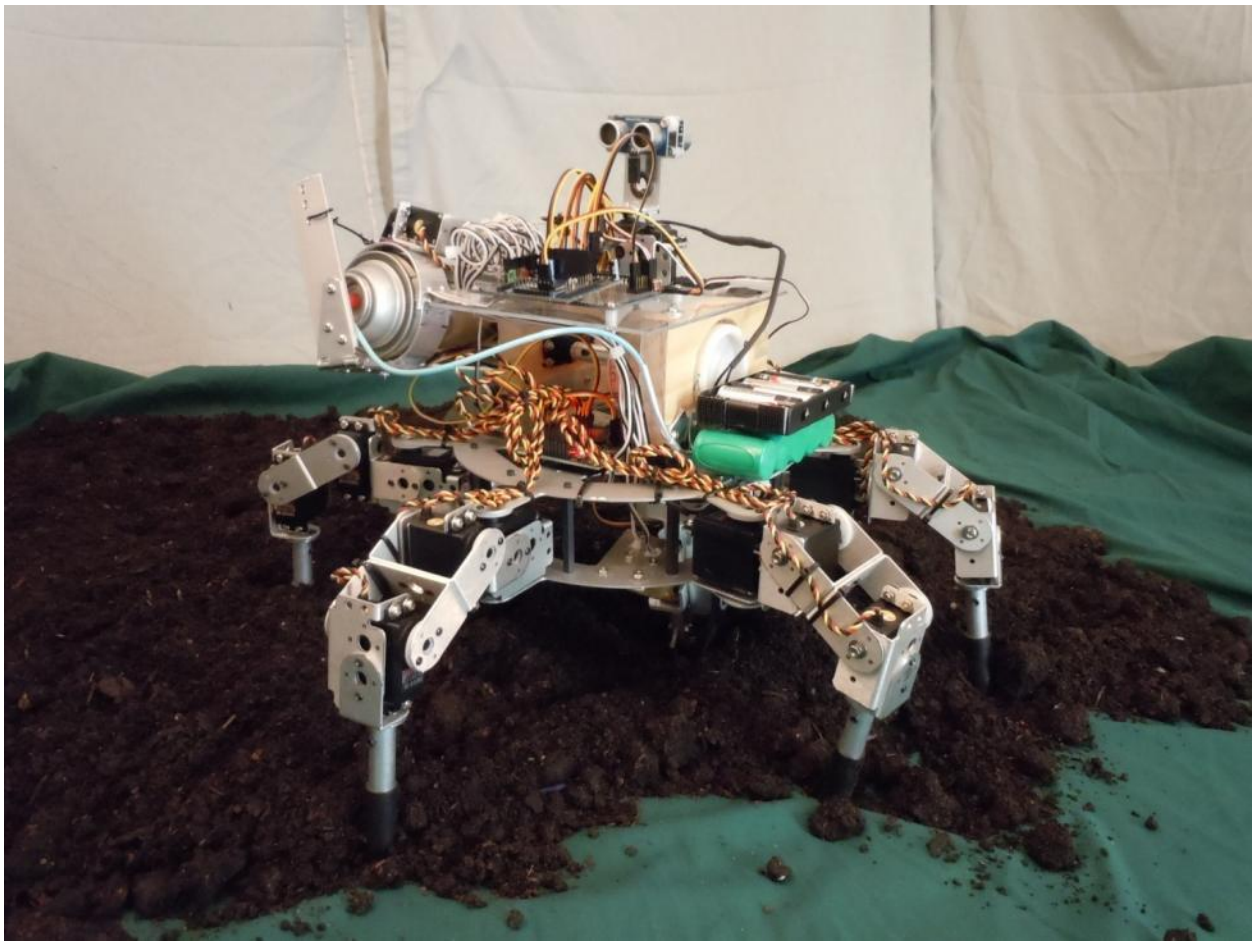
Overhead View showing Propeller Schmart Board and Parallax Prototyping Board



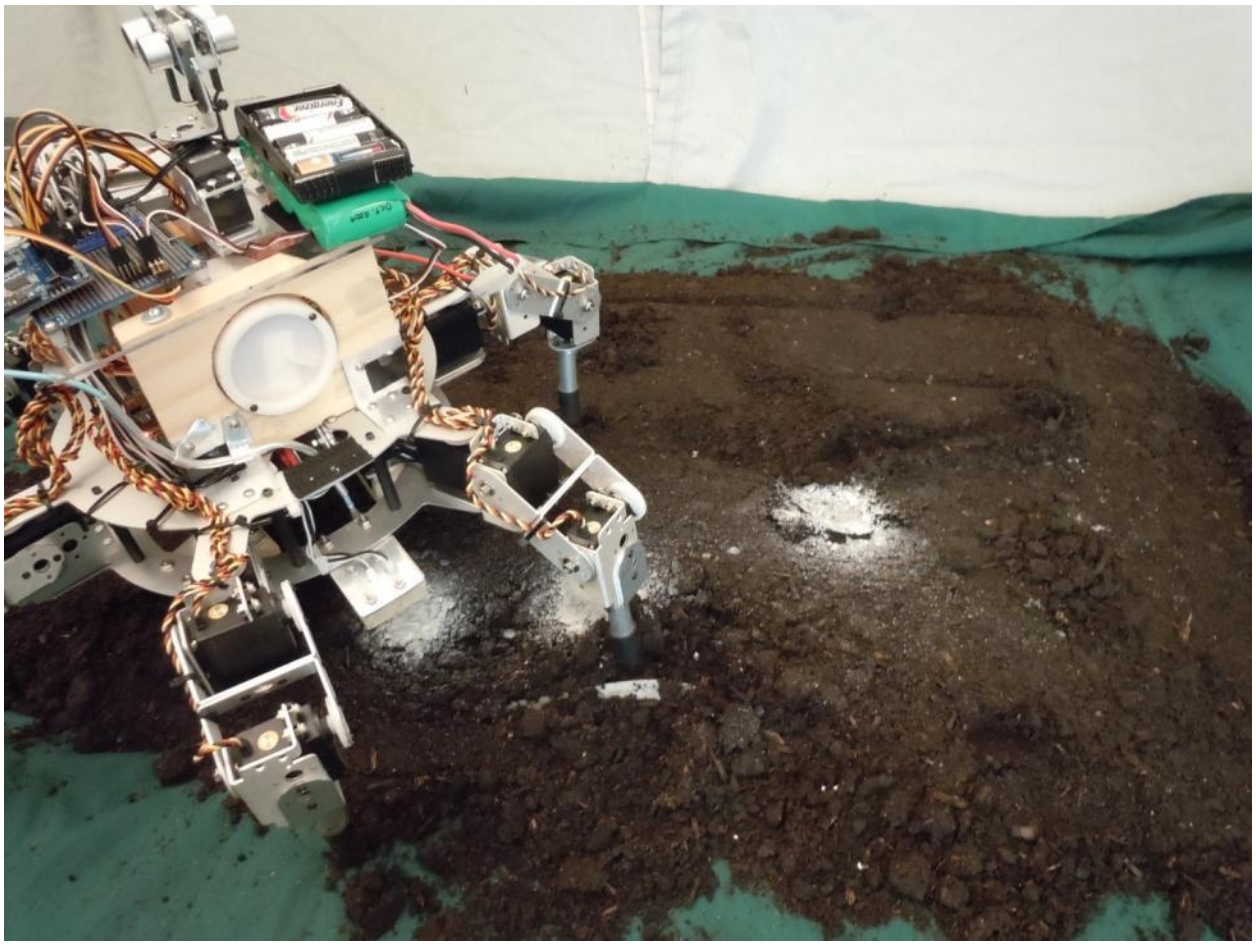
Overhead View showing Propeller Schmart Board and Parallax Prototyping Board



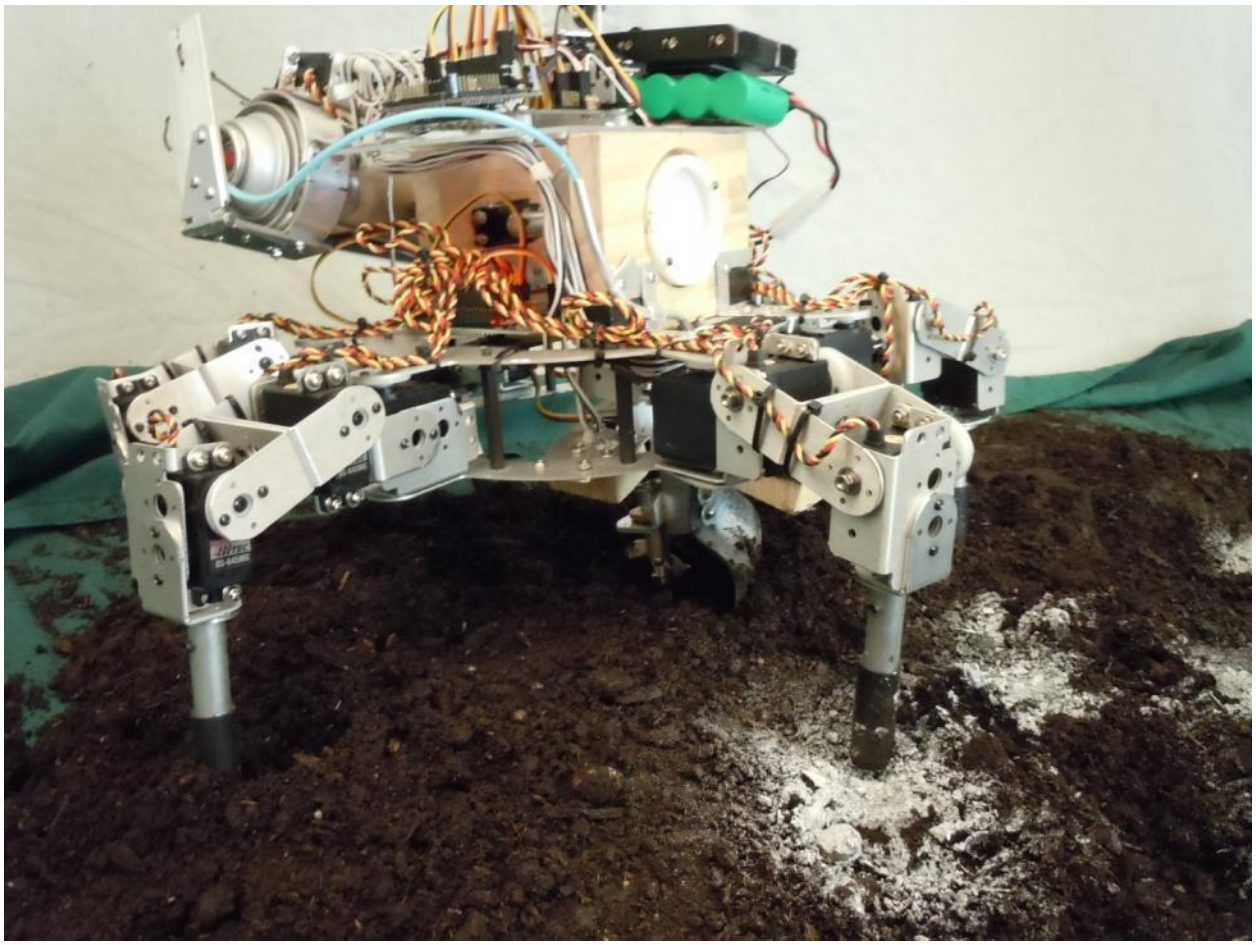
Side View Showing One of the Two Parallax Propeller Servo Controllers



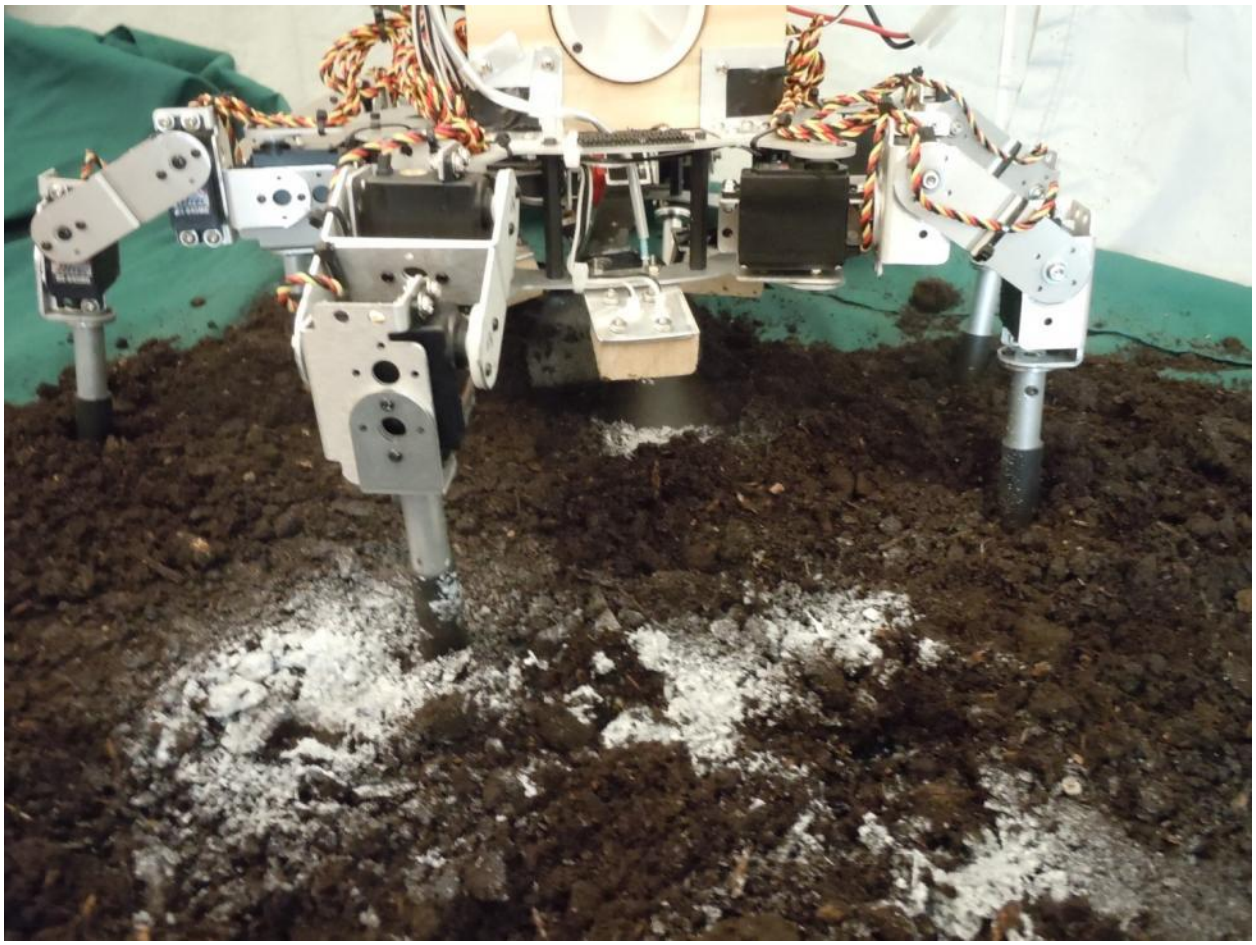
Prospero Walking Across Demo Field (The ground is frozen; It's December in Iowa, USA)



Prospero starting to plant seeds. The biodegradable paint marks the spots



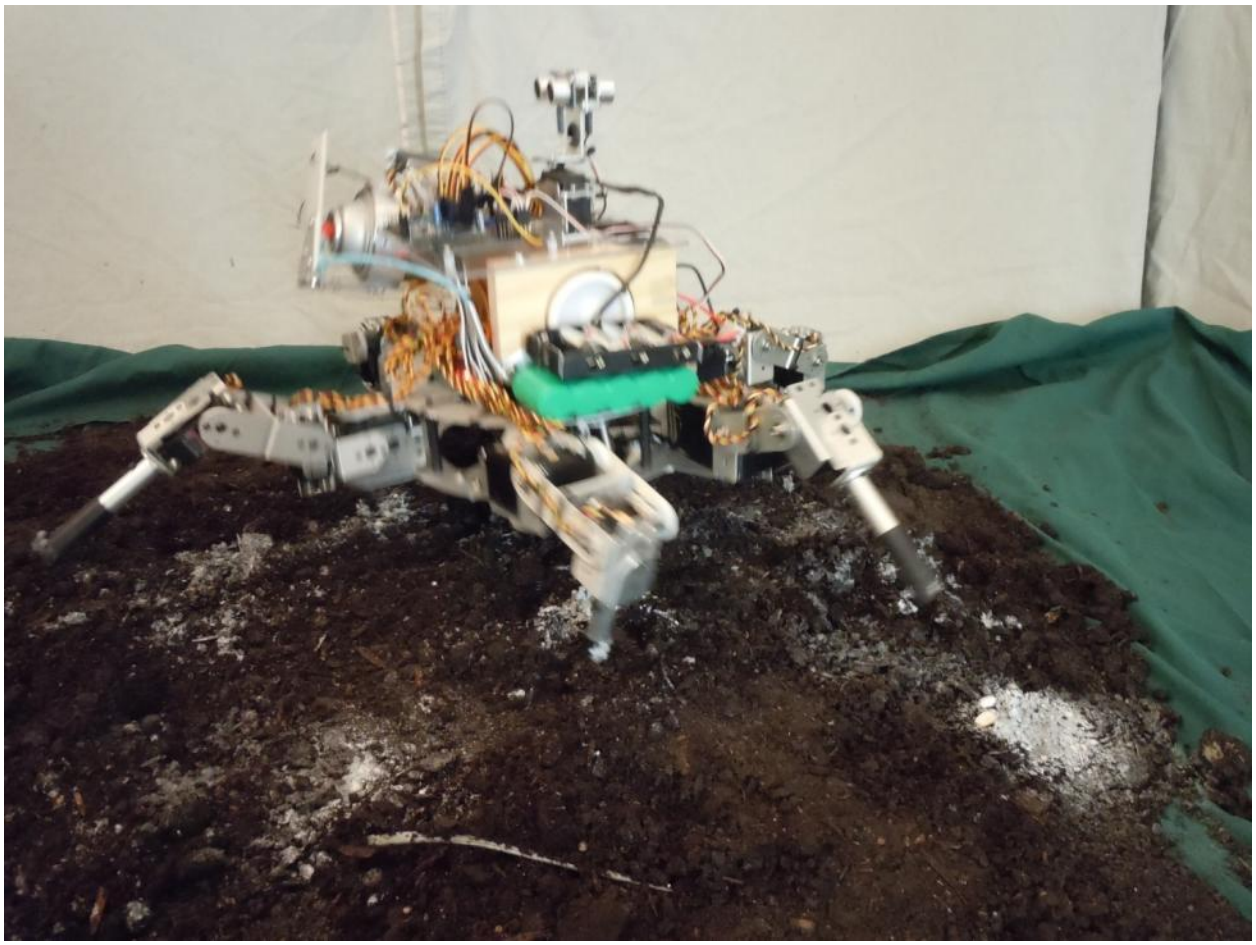
Prospero with Seed Driller Extended, Drilling hole



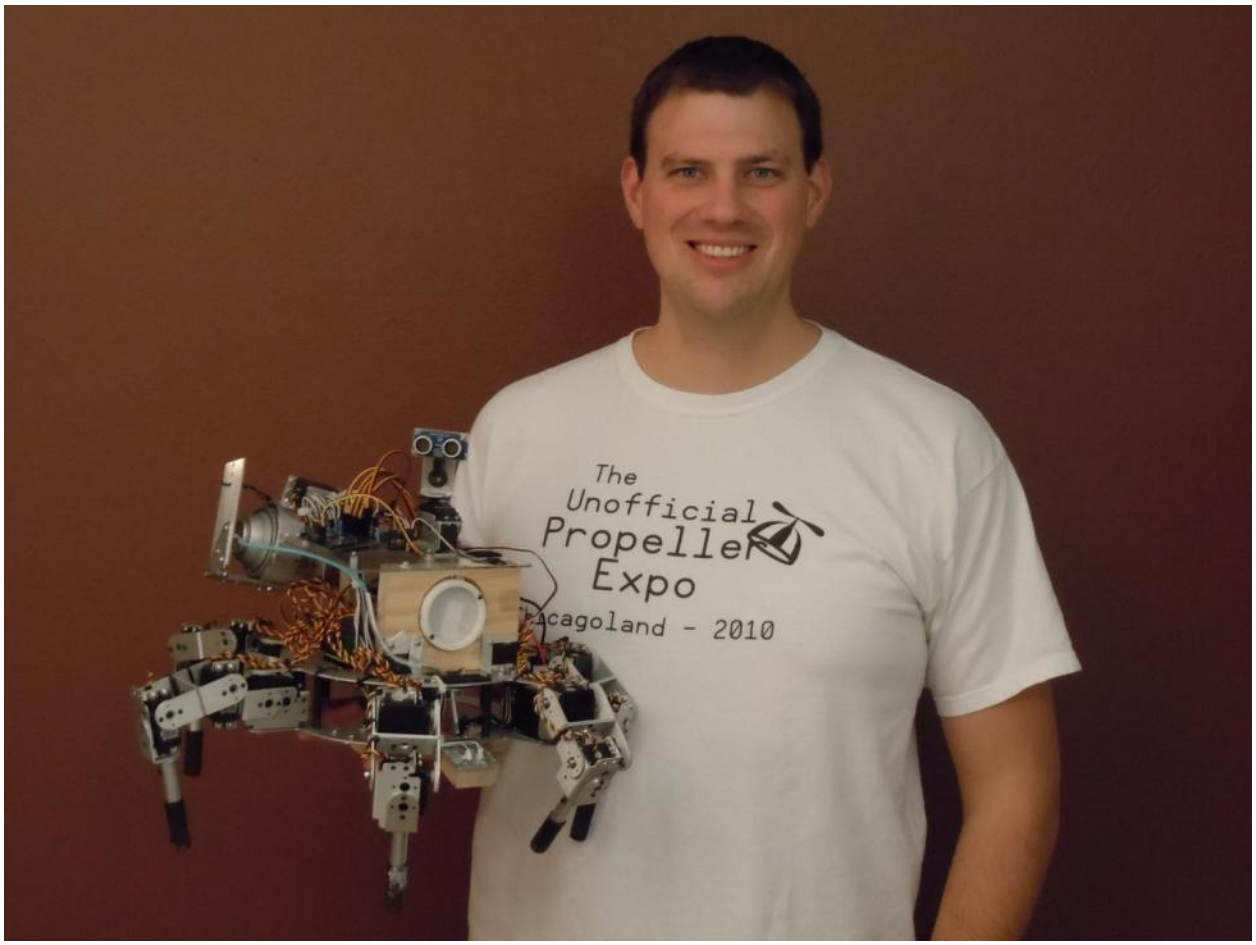
Prospero Spraying Biodegradable Paint, Marking Planted Seed

Planted Seeds

Dug-up Planted Seed



Prospero Walking to New Spot After Detecting that a Seed was Planted there



Author and Builder with Prospero

Source Code

Listed below is the source code used in this project.

Objects shipped with the Parallax Propeller Tool and those found on the OBEX have not been included for clarity.

```
'Prospero: Robotic Farmer (C) CopyRight 2010 All rights reserved
'David Dorhout vanmunch@yahoo.com
'
'
'Works for looking forward in any direction.
'
'***This program walks, avaoids obsticals, detects planted seeds, plants seeds,
'marks them in full 360***
CON

_CLKMODE        = XTAL1 + PLL16X
_XINFREQ        = 5_000_000

'Constants for using the Propeller Servo Control (PSC) boards
COMPIN          = 23                'Pin used for communication with the PSC
PSC_BAUD        = 0                 'Baud rate (0 - 2400, 1 - 38400)
Ramp            = 10                'Ramp is the speed between 0-63 that the PSC turns
Ramp2           = 60                'the servos  (fast to slow)
Ramp3           = 30


'Constants for the 18 servos used in the hips, legs, and feet; "a" is for the 0-180
'degrees of motion and  the "b" is for the 181-360 degrees of motion
'"a" side************
fRPawFoot       = 31
fRPawKnee       = 30
fRPawHip        = 29

fRFFoot         = 26
fRFKnee         = 27
fRFHip          = 28

fRRFoot         = 23
fRRKnee         = 22
fRRHip          = 21

fLPawFoot       =  0
fLPawKnee       =  1
fLPawHip        =  2

fLFFoot         = 15
fLFKnee         = 14
fLFHip          = 13

fLRFoot         =  3
fLRKnee         =  4
fLRHip          =  5

'"b" side************
bRPawFoot       =  3 '31
bRPawKnee       =  4 '30
```

```
bRPawFoot         =  3 '31
bRPawKnee         =  4 '30
bRPawHip          =  5 '29

bRFFoot           = 15 '26
bRFKnee           = 14 '27
bRFHip            = 13 '28

bRRFoot           =  0 '23
bRRKnee           =  1 '22
bRRHip            =  2 '21

bLPawFoot         = 23  '0
bLPawKnee         = 22  '1
bLPawHip          = 21  '2

bLFFoot           = 26 '15
bLFKnee           = 27 '14
bLFHip            = 28 '13

bLRFoot           = 31  '3
bLRKnee           = 30  '4
bLRHip            = 29  '5


'Ping)))
fPING_Pin         = 22    'I/O Pin For PING)))
bPING_Pin         = 21
PingServoM        = 6     'connected via PSC

'Seed dispenser
SeedDispenserLED  = 1
SeedDispenserPR   = 0

'Planted seed dector array
SeedMarkerServo   = 7     'connected via PSC
SeedDectorPR1     = 2
SeedDectorPR2     = 3
SeedDectorPR3     = 4
SeedDectorLEDS    = 5

'Seed driller
TubeServo         = 25
DrillServo        = 24

SeedServo         = 20

VAR

'Variables for the planted seed dector array
Long                  SeedDectorPR1time
```

```
'Variables for the planted seed dector array
Long                    SeedDectorPR1time
Long                    SeedDectorPR2time
Long                    SeedDectorPR3time
Long                    SeedDectorPR1timeThreshold
Long                    SeedDectorPR2timeThreshold
Long                    SeedDectorPR3timeThreshold

'Variables for the seed droper
Long                    photoresistorBase
Long                    photoresistorNew
Long                    photoresistorThreshhold
Long                    time
Long                    timeold

'Variables for the Ping)))
Long                    range             'Distance for Ping)))
Long                    PingServoD        'Direction/PW of the servo holding the Ping)))
Long                    PingServoR
Long                    PingServoL
Long                    DirectionPW

Long                    RPawFootPW
Long                    RPawKneePW
Long                    RPawHipPW

Long                    RFFootPW
Long                    RFKneePW
Long                    RFHipPW

Long                    RRFootPW
Long                    RRKneePW
Long                    RRHipPW

Long                    LPawFootPW
Long                    LPawKneePW
Long                    LPawHipPW

Long                    LFFootPW
Long                    LFKneePW
Long                    LFHipPW

Long                    LRFootPW
Long                    LRKneePW
Long                    LRHipPW

Long                    RPawFtSign
Long                    RFFtSign
Long                    RRFtSign

Long                    PW
```

```
Long                    PW

OBJ
     PSC    : "ServoControllerSerial"
    ping    : "ping"
   Debug    : "FullDuplexSerialPlus"
   fmath    : "FloatMath"
   Fstring  : "FloatString"

PUB Start
  PSC.START(COMPIN, PSC_BAUD)
  Debug.Start(31, 30, 0, 57600)

  Intialize

PUB Intialize

  PW := 750
  PSC.SETPOS(SeedServo, Ramp, PW)             'Servo that dispenses the seeds
                                              'Keeps the digging unit retracted 370 is
  PSC.SETPOS(TubeServo, Ramp, 360)            ' as close to the top as it can be
                                              '450 is the close as it can be if the bit
  'Set leg home position                       'catches the bottom
  PW := 750
  PSC.SETPOS(fLPawKnee, Ramp2, PW)
  PSC.SETPOS(fRFKnee, Ramp2, PW)
  PSC.SETPOS(fLRKnee, Ramp2, PW)

  PSC.SETPOS(fRPawKnee, Ramp2, PW)
  PSC.SETPOS(fLFKnee, Ramp2, PW)
  PSC.SETPOS(fRRKnee, Ramp2, PW)


  PW := 750
  PSC.SETPOS(fLPawFoot, Ramp2, PW)
  PSC.SETPOS(fRFFoot, Ramp2, PW)
  PSC.SETPOS(fLRFoot, Ramp2, PW)

  PSC.SETPOS(fRPawFoot, Ramp2, PW)
  PSC.SETPOS(fLFFoot, Ramp2, PW)
  PSC.SETPOS(fRRFoot, Ramp2, PW)

  waitcnt(clkfreq*5 + cnt)                    '5 second wait


'Take baseline measurements of the three photorisistors and caculate what the
'threshold is for deciding if there is a paint spot (seed planted) there

'waitcnt(clkfreq*5 + cnt)                    'For debuging
  dira[SeedDectorLEDS] := 1
```

```
  dira[SeedDectorLEDS] := 1
  outa[SeedDectorLEDS] := 1
  waitcnt(clkfreq/30 + cnt)                 'Turns the LEDS on and lets them get ready
                                            'for photoresister

'SeedDectorPR1time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR1             'set APIN to SeedDectorPR1 (PSeedDectorPR1)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR1] := outa[SeedDectorPR1] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge


    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR1]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR1time := (phsa - 624) #> 0
    SeedDectorPR1timeThreshold :=  fmath.fMul(SeedDectorPR1time, 0.8)

    'Display results
    debug.str(string(13, "SeedDectorPR1time = "))
    debug.dec(SeedDectorPR1time)
    debug.str(string(13, "SeedDectorPR1timeThreshold = "))
    debug.dec(SeedDectorPR1timeThreshold)


'SeedDectorPR2time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR2             'set APIN to SeedDectorPR2 (PSeedDectorPR2)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR2] := outa[SeedDectorPR2] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge


    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR2]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR2time := (phsa - 624) #> 0
    SeedDectorPR2timeThreshold :=  fmath.fMul(SeedDectorPR2time, 0.8)
```

```
    SeedDectorPR2time := (phsa - 624) #> 0
    SeedDectorPR2timeThreshold :=  fmath.fMul(SeedDectorPR2time, 0.8)

    'Display results
    debug.str(string(13, "SeedDectorPR2time = "))
    debug.dec(SeedDectorPR2time)
    debug.str(string(13, "SeedDectorPR2timeThreshold = "))
    debug.dec(SeedDectorPR2timeThreshold)

'SeedDectorPR3time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR3             'set APIN to SeedDectorPR3 (PSeedDectorPR3)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR3] := outa[SeedDectorPR3] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge


    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR3]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR3time := (phsa - 624) #> 0
    SeedDectorPR3timeThreshold :=  fmath.fMul(SeedDectorPR3time, 0.8)

    'Display results
    debug.str(string(13, "SeedDectorPR3time = "))
    debug.dec(SeedDectorPR3time)
    debug.str(string(13, "SeedDectorPR3timeThreshold = "))
    debug.dec(SeedDectorPR3timeThreshold)

    waitcnt(clkfreq*2 + cnt)
  outa[SeedDectorLEDS] := 0                 'Turns LEDs off

  MainProgramf
PUB MainProgramf

'Check For Planted Seed*************************************************************
'************************************************************************************

'waitcnt(clkfreq*3 + cnt)                   'For debuging
  dira[SeedDectorLEDS] := 1
  outa[SeedDectorLEDS] := 1
  waitcnt(clkfreq/30 + cnt)                 'Turns the LEDS on and lets them get ready fo
```

```
  outa[SeedDectorLEDS] := 1
  waitcnt(clkfreq/30 + cnt)                 'Turns the LEDS on and lets them get ready for photoresister


'SeedDectorPR1time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR1             'set APIN to SeedDectorPR1 (PSeedDectorPR1)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR1] := outa[SeedDectorPR1] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR1]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR1time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR1time = "))
    debug.dec(SeedDectorPR1time)

'SeedDectorPR2time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR2             'set APIN to SeedDectorPR2 (PSeedDectorPR2)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR2] := outa[SeedDectorPR2] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR2]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR2time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR2time = "))
    debug.dec(SeedDectorPR2time)

'SeedDectorPR3time*********
```

```
    debug.dec(SeedDectorPR2time)

'SeedDectorPR3time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR3             'set APIN to SeedDectorPR3 (PSeedDectorPR3)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR3] := outa[SeedDectorPR3] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR3]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR3time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR3time = "))
    debug.dec(SeedDectorPR3time)

    waitcnt(clkfreq*2 + cnt)
    outa[SeedDectorLEDS] := 0               'Turns LEDs off

    'if the new time is below the threshold then it is detecting a planted seed and
    'will go to the walking program sequence
    IF SeedDectorPR1time =< SeedDectorPR1timeThreshold
      fStart_And_Intialize_Variables
    IF SeedDectorPR2time =< SeedDectorPR2timeThreshold
      fStart_And_Intialize_Variables
    IF SeedDectorPR3time =< SeedDectorPR3timeThreshold
      fStart_And_Intialize_Variables

'Digging seed hole*************************************************************************
'*******************************************************************************************


  PW := 550
  PSC.SETPOS(fLPawKnee, Ramp3, PW)
  PSC.SETPOS(fRFKnee, Ramp3, PW)
  PSC.SETPOS(fLRKnee, Ramp3, PW)

  PSC.SETPOS(fRPawKnee, Ramp3, PW)
  PSC.SETPOS(fLFKnee, Ramp3, PW)
  PSC.SETPOS(fRRKnee, Ramp3, PW)
```

```
PSC.SETPOS(fRRKnee, Ramp3, PW)


PW := 550
PSC.SETPOS(fLPawFoot, Ramp3, PW)
PSC.SETPOS(fRFFoot, Ramp3, PW)
PSC.SETPOS(fLRFoot, Ramp3, PW)

PSC.SETPOS(fRPawFoot, Ramp3, PW)
PSC.SETPOS(fLFFoot, Ramp3, PW)
PSC.SETPOS(fRRFoot, Ramp3, PW)
waitcnt(clkfreq*2 + cnt)


PSC.SETPOS(TubeServo, Ramp, 850)            'Planting mechanisum fully extended
waitcnt(clkfreq*1 + cnt)
PSC.SETPOS(TubeServo, Ramp, 750)            'Planting mechanisum stright down
waitcnt(clkfreq*1 + cnt)

PSC.SETPOS(DrillServo, Ramp, 1300)          'Drill on



PW := 750
PSC.SETPOS(fLPawKnee, Ramp2, PW)
PSC.SETPOS(fRFKnee, Ramp2, PW)
PSC.SETPOS(fLRKnee, Ramp2, PW)

PSC.SETPOS(fRPawKnee, Ramp2, PW)
PSC.SETPOS(fLFKnee, Ramp2, PW)
PSC.SETPOS(fRRKnee, Ramp2, PW)


PW := 750
PSC.SETPOS(fLPawFoot, Ramp2, PW)
PSC.SETPOS(fRFFoot, Ramp2, PW)
PSC.SETPOS(fLRFoot, Ramp2, PW)

PSC.SETPOS(fRPawFoot, Ramp2, PW)
PSC.SETPOS(fLFFoot, Ramp2, PW)
PSC.SETPOS(fRRFoot, Ramp2, PW)

waitcnt(clkfreq*12 + cnt)

PW := 600
PSC.SETPOS(fLPawKnee, Ramp3, PW)
PSC.SETPOS(fRFKnee, Ramp3, PW)
PSC.SETPOS(fLRKnee, Ramp3, PW)

PSC.SETPOS(fRPawKnee, Ramp3, PW)
```

```
  PSC.SETPOS(fLFKnee, Ramp3, PW)
  PSC.SETPOS(fRRKnee, Ramp3, PW)


  PW := 600
  PSC.SETPOS(fLPawFoot, Ramp3, PW)
  PSC.SETPOS(fRFFoot, Ramp3, PW)
  PSC.SETPOS(fLRFoot, Ramp3, PW)

  PSC.SETPOS(fRPawFoot, Ramp3, PW)
  PSC.SETPOS(fLFFoot, Ramp3, PW)
  PSC.SETPOS(fRRFoot, Ramp3, PW)

  PSC.SETPOS(TubeServo, Ramp, 750)
  PSC.SETPOS(DrillServo, Ramp, 750)


'Drop Seed*************************************************************************************
'**********************************************************************************************
'Seed dispenser
'SeedDispenserLED = 1
'SeedDispenserPR  = 0

  'Configure counter module for seed detector.

  ctra[30..26] := %01000                        'Set mode to "POS detector"
  ctra[5..0]   := SeedDispenserPR               'set APIN to SeedDispenserPR
  frqa := 1                                     '(SeedDispenserPR)


  'Turning on the LED light
  dira[SeedDispenserLED] := 1
  outa[SeedDispenserLED] := 1
  waitcnt(clkfreq/20 + cnt)                     'Let the LED turn on

  'Using the photoresistor to detect a seed droping
  'repeat
    'Charge RC circuit.
    dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                          'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                       'Clear the phsa register
    dira[SeedDispenserPR]~                      'Pin to input stops charging circuit
      waitcnt(clkfreq/60 + cnt)
    time := (phsa - 624) #> 0
    photoresistorBase := time
    photoresistorThreshhold := photoresistorBase /5 + photoresistorBase

    'Display results
    debug.str(string(13, "time = "))
```

```
'Display results
debug.str(string(13, "time = "))
debug.dec(time)
debug.str(string(13, "photoresistorBase = "))
debug.dec(photoresistorBase)
debug.str(string(13, "photoresistorThreshhold = "))
debug.dec(photoresistorThreshhold)

repeat
  PW := 350
  PSC.SETPOS(SeedServo, Ramp, PW)
  repeat 60
   Charge RC circuit.
    dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1 'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge


    phsa~                                 'Clear the phsa register
    dira[SeedDispenserPR]~                'Pin to input stops charging circuit
      waitcnt(clkfreq/60 + cnt)
    time := (phsa - 624) #> 0
    IF time => timeold
      timeold := time
      debug.str(string(13, "timeold = "))
      debug.dec(timeold)
    If timeold => photoresistorThreshhold
      debug.str(string(13, "TimeOld is above thresh hold!"))
       outa[SeedDispenserLED] := 0
       CoverAndMarkSeedPlacementF

'     waitcnt(clkfreq*1 + cnt)

  PW := 1150
  PSC.SETPOS(SeedServo, Ramp, PW)
  repeat 60
   'Charge RC circuit.
    dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1 'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge


    phsa~                                 'Clear the phsa register
    dira[SeedDispenserPR]~                'Pin to input stops charging circuit
      waitcnt(clkfreq/60 + cnt)
    time := (phsa - 624) #> 0
    IF time => timeold
      timeold := time
      debug.str(string(13, "timeold = "))
      debug.dec(timeold)
    If timeold => photoresistorThreshhold
       debug.str(string(13, "TimeOld is above thresh hold!"))
       outa[SeedDispenserLED] := 0
```

```
            debug.str(string(13, "TimeOld is above thresh hold!"))
            outa[SeedDispenserLED] := 0
            CoverAndMarkSeedPlacementF

PUB MainProgramB

'Check For Planted Seed*************************************************************************
'************************************************************************************************

'waitcnt(clkfreq*3 + cnt)                    'For debuging
  dira[SeedDectorLEDS] := 1
  outa[SeedDectorLEDS] := 1
  waitcnt(clkfreq/30 + cnt)                 'Turns the LEDS on and lets them get ready


'SeedDectorPR1time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR1             'set APIN to SeedDectorPR1 (PSeedDectorPR1)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR1] := outa[SeedDectorPR1] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR1]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR1time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR1time = "))
    debug.dec(SeedDectorPR1time)

'SeedDectorPR2time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR2             'set APIN to SeedDectorPR2 (PSeedDectorPR2)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR2] := outa[SeedDectorPR2] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
```

```
    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR2]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR2time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR2time = "))
    debug.dec(SeedDectorPR2time)

'SeedDectorPR3time*********
    'Configure counter module.

    ctra[30..26] := %01000                  'Set mode to "POS detector"
    ctra[5..0] := SeedDectorPR3             'set APIN to SeedDectorPR3 (PSeedDectorPR3)
    frqa := 1

    'Charge RC circuit.
    dira[SeedDectorPR3] := outa[SeedDectorPR3] := 1     'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                      'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                   'Clear the phsa register
    dira[SeedDectorPR3]~                    'Pin to input stops charging circuit

      waitcnt(clkfreq/60 + cnt)

    SeedDectorPR3time := (phsa - 624) #> 0

    'Display results
    debug.str(string(13, "SeedDectorPR3time = "))
    debug.dec(SeedDectorPR3time)

    waitcnt(clkfreq*2 + cnt)
    outa[SeedDectorLEDS] := 0               'Turns LEDs off

    'if the new time is below the threshold then it is detecting a planted seed and
    'will go to the walking program sequence
    IF SeedDectorPR1time =< SeedDectorPR1timeThreshold
      bStart_And_Intialize_Variables
    IF SeedDectorPR2time =< SeedDectorPR2timeThreshold
      bStart_And_Intialize_Variables
    IF SeedDectorPR3time =< SeedDectorPR3timeThreshold
      bStart_And_Intialize_Variables

'Digging seed hole*************************************************************************
'******************************************************************************************
```

```
'*****************************************************************************************


  PW := 550
  PSC.SETPOS(bLPawKnee, Ramp3, PW)
  PSC.SETPOS(bRFKnee, Ramp3, PW)
  PSC.SETPOS(bLRKnee, Ramp3, PW)

  PSC.SETPOS(bRPawKnee, Ramp3, PW)
  PSC.SETPOS(bLFKnee, Ramp3, PW)
  PSC.SETPOS(bRRKnee, Ramp3, PW)


  PW := 550
  PSC.SETPOS(bLPawFoot, Ramp3, PW)
  PSC.SETPOS(bRFFoot, Ramp3, PW)
  PSC.SETPOS(bLRFoot, Ramp3, PW)

  PSC.SETPOS(bRPawFoot, Ramp3, PW)
  PSC.SETPOS(bLFFoot, Ramp3, PW)
  PSC.SETPOS(bRRFoot, Ramp3, PW)
  waitcnt(clkfreq*2 + cnt)


  PSC.SETPOS(TubeServo, Ramp, 850)            'Planting mechanisum fully extended
  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 750)            'Planting mechanisum stright down
  waitcnt(clkfreq*1 + cnt)

  PSC.SETPOS(DrillServo, Ramp, 1300)          'Drill on



  PW := 750
  PSC.SETPOS(bLPawKnee, Ramp2, PW)
  PSC.SETPOS(bRFKnee, Ramp2, PW)
  PSC.SETPOS(bLRKnee, Ramp2, PW)

  PSC.SETPOS(bRPawKnee, Ramp2, PW)
  PSC.SETPOS(bLFKnee, Ramp2, PW)
  PSC.SETPOS(bRRKnee, Ramp2, PW)


  PW := 750
  PSC.SETPOS(bLPawFoot, Ramp2, PW)
  PSC.SETPOS(bRFFoot, Ramp2, PW)
  PSC.SETPOS(bLRFoot, Ramp2, PW)

  PSC.SETPOS(bRPawFoot, Ramp2, PW)
  PSC.SETPOS(bLFFoot, Ramp2, PW)
  PSC.SETPOS(bRRFoot, Ramp2, PW)
```

```
  PSC.SETPOS(bRRFoot, Ramp2, PW)

  waitcnt(clkfreq*12 + cnt)

  PW := 600
  PSC.SETPOS(bLPawKnee, Ramp3, PW)
  PSC.SETPOS(bRFKnee, Ramp3, PW)
  PSC.SETPOS(bLRKnee, Ramp3, PW)

  PSC.SETPOS(bRPawKnee, Ramp3, PW)
  PSC.SETPOS(bLFKnee, Ramp3, PW)
  PSC.SETPOS(bRRKnee, Ramp3, PW)


  PW := 600
  PSC.SETPOS(bLPawFoot, Ramp3, PW)
  PSC.SETPOS(bRFFoot, Ramp3, PW)
  PSC.SETPOS(bLRFoot, Ramp3, PW)

  PSC.SETPOS(bRPawFoot, Ramp3, PW)
  PSC.SETPOS(bLFFoot, Ramp3, PW)
  PSC.SETPOS(bRRFoot, Ramp3, PW)

  PSC.SETPOS(TubeServo, Ramp, 750)
  PSC.SETPOS(DrillServo, Ramp, 750)


'Drop Seed*******************************************************************************
'*****************************************************************************************
'Seed dispenser
'SeedDispenserLED = 1
'SeedDispenserPR  = 0

  'Configure counter module for seed detector.

  ctra[30..26] := %01000                        'Set mode to "POS detector"
  ctra[5..0]   := SeedDispenserPR               'set APIN to SeedDispenserPR (SeedDispenserP
  frqa := 1


  'Turning on the LED light
  dira[SeedDispenserLED] := 1
  outa[SeedDispenserLED] := 1
  waitcnt(clkfreq/20 + cnt)                     'Let the LED turn on

  'Using the photoresistor to detect a seed droping
  'repeat
    'Charge RC circuit.
    dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1       'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                            'Wait for circuit to charge
```

```
    'Charge RC circuit.
    dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1       'Set pin to output-high
    waitcnt(clkfreq/100_000 + cnt)                            'Wait for circuit to charge

    'Start RC decay measurement.
    phsa~                                     'Clear the phsa register
    dira[SeedDispenserPR]~                    'Pin to input stops charging circuit
      waitcnt(clkfreq/60 + cnt)
    time := (phsa - 624) #> 0
    photoresistorBase := time
    photoresistorThreshhold := photoresistorBase /5 + photoresistorBase

    'Display results
    debug.str(string(13, "time = "))
    debug.dec(time)
    debug.str(string(13, "photoresistorBase = "))
    debug.dec(photoresistorBase)
    debug.str(string(13, "photoresistorThreshhold = "))
    debug.dec(photoresistorThreshhold)

    repeat
      PW := 350
      PSC.SETPOS(SeedServo, Ramp, PW)
      repeat 60
        Charge RC circuit.
        dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1   'Set pin to output-high
        waitcnt(clkfreq/100_000 + cnt)                        'Wait for circuit to charge


        phsa~                                 'Clear the phsa register
        dira[SeedDispenserPR]~                'Pin to input stops charging circuit
          waitcnt(clkfreq/60 + cnt)
        time := (phsa - 624) #> 0
        IF time => timeold
          timeold := time
          debug.str(string(13, "timeold = "))
          debug.dec(timeold)
        If timeold => photoresistorThreshhold
          debug.str(string(13, "TimeOld is above thresh hold!"))
          outa[SeedDispenserLED] := 0
          CoverAndMarkSeedPlacementB

'       waitcnt(clkfreq*1 + cnt)

      PW := 1150
      PSC.SETPOS(SeedServo, Ramp, PW)
      repeat 60
        'Charge RC circuit.
        dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1   'Set pin to output-high
        waitcnt(clkfreq/100_000 + cnt)                        'Wait for circuit to charge
```

```
      dira[SeedDispenserPR] := outa[SeedDispenserPR] := 1   'Set pin to output-high
      waitcnt(clkfreq/100_000 + cnt)                        'Wait for circuit to charge


      phsa~                                 'Clear the phsa register
      dira[SeedDispenserPR]~                'Pin to input stops charging circuit
        waitcnt(clkfreq/60 + cnt)
      time := (phsa - 624) #> 0
      IF time => timeold
        timeold := time
        debug.str(string(13, "timeold = "))
        debug.dec(timeold)
      If timeold => photoresistorThreshhold
          debug.str(string(13, "TimeOld is above thresh hold!"))
          outa[SeedDispenserLED] := 0
          CoverAndMarkSeedPlacementB


PUB CoverAndMarkSeedPlacementF

'Cover Seed and Mark with Paint*****************************************************
'***********************************************************************************
  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 450)

  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 850)

  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 450)


'Spray paint using the SeedMarkerServo

  PSC.SETPOS(SeedMarkerServo, Ramp, 850)      'Spray paint
  waitcnt(clkfreq*3 + cnt)                    'for 3 seconds
  PSC.SETPOS(SeedMarkerServo, Ramp, 750)      'Turn off sprayer


  fStart_And_Intialize_Variables

PUB CoverAndMarkSeedPlacementB

'Cover Seed and Mark with Paint*****************************************************
'***********************************************************************************
  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 450)

  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 850)
```

```
  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 850)

  waitcnt(clkfreq*1 + cnt)
  PSC.SETPOS(TubeServo, Ramp, 450)


'Spray paint using the SeedMarkerServo

  PSC.SETPOS(SeedMarkerServo, Ramp, 850)      'Spray paint
  waitcnt(clkfreq*3 + cnt)                    'for 3 seconds
  PSC.SETPOS(SeedMarkerServo, Ramp, 750)      'Turn off sprayer


  bStart_And_Intialize_Variables
PUB fStart_And_Intialize_Variables

  debug.str(string(13, "fStart_And_Intialize_Variables :) "))

'Intialize variables
  PingServoD := 751
  Debug.dec(PingServoD)
  fPathCheck

PUB bStart_And_Intialize_Variables

'Intialize variables
  PingServoD := 751
  Debug.dec(PingServoD)
  bPathCheck

PUB fPathCheck   'Use the fPing))) to chekc to see if the path is clear
  'waitcnt(clkfreq*4 + cnt)
  'Debug.dec(PingServoD)
   Debug.Str(String(13, "fPathCheck"))

     PingServoL := PingServoD
     PingServoR := PingServoD

  Repeat
   PingServoL*****************************************************************************
   IF PingServoL => 1150
        Debug.Str(String(13, "Left Side is compleatly Blocked!"))
        bStart_And_Intialize_Variables
   PSC.SETPOS(PingServoM, 0, PingServoL)
   waitcnt(clkfreq/2 + cnt)                   'Gives the PingServo moter time to move

    range := ping.Inches(fPING_Pin)           'Get Range In Inches
    'Debug.Str(String(13, "Is the Ping working?"))  '13 gives a charage return
    Debug.tx(Debug#CR)                                'Gives a charage return
```

```
    Debug.tx(Debug#CR)                              'Gives a charage return
    Debug.dec(range)                                'Gives the distance is inches via
    'waitcnt(clkfreq / 10 + cnt)                    'the Ping)))
    IF range => 23
      PingServoD := PingServoL
      Debug.Str(String("Hexapod is walking!"))
      fHexapodWalking                               'Goes to Pub "HexapodWalking" to start
    ELSE                                            'walking
      Debug.Str(String(13, "Left Blocked!"))
      PingServoL := PingServoL+75


  PingServoR*****************************************************************************
  IF PingServoL =< 350
      Debug.Str(String(13, "Right Side is compleatly Blocked!"))
      bStart_And_Intialize_Variables
  PSC.SETPOS(PingServoM, 0, PingServoR)
  waitcnt(clkfreq/2 + cnt)                          'Gives the PingServo moter time to move

    range := ping.Inches(fPING_Pin)                 'Get Range In Inches
    'Debug.Str(String(13, "Is the Ping working?"))  '13 gives a charage return
    Debug.tx(Debug#CR)                              'Gives a charage return
    Debug.dec(range)                                'Gives the distance is inches via
    'waitcnt(clkfreq / 10 + cnt)                    'the Ping)))
    IF range => 23
      PingServoD := PingServoR
      Debug.Str(String("Hexapod is walking!"))
      fHexapodWalking                               'Goes to Pub "HexapodWalking" to start
    ELSE                                            'walking
      Debug.Str(String(13, "Left Blocked!"))
      PingServoR := PingServoR-75

PUB bPathCheck   'Use the bPing))) to chekc to see if the path is clear
  'waitcnt(clkfreq*4 + cnt)
  'Debug.dec(PingServoD)
   Debug.Str(String(13, "bPathCheck"))

     PingServoL := PingServoD
     PingServoR := PingServoD

  Repeat
   PingServoL*****************************************************************************
   IF PingServoL =< 350 '=> 1150
      Debug.Str(String(13, "Left Side is compleatly Blocked!"))
      fStart_And_Intialize_Variables
   PSC.SETPOS(PingServoM, 0, PingServoL)
   waitcnt(clkfreq/2 + cnt)                         'Gives the PingServo moter time to move

    range := ping.Inches(bPING_Pin)                 'Get Range In Inches
    'Debug.Str(String(13, "Is the Ping working?"))  '13 gives a charage return
    Debug.tx(Debug#CR)                              'Gives a charage return
```

```
    Debug.tx(Debug#CR)                              'Gives a charage return
    Debug.dec(range)                                'Gives the distance is inches via
    'waitcnt(clkfreq / 10 + cnt)                    'the Ping)))
    IF range => 23
      PingServoD := PingServoL
      Debug.Str(String("Hexapod is walking!"))
      bHexapodWalking                           'Goes to Pub "HexapodWalking" to start
    ELSE                                        'walking
      Debug.Str(String(13, "Left Blocked!"))
      PingServoL := PingServoL-75 '+75


  PingServoR*****************************************************************************
  IF PingServoL => 1150 '=< 350
      Debug.Str(String(13, "Right Side is compleatly Blocked!"))
      fStart_And_Intialize_Variables
  PSC.SETPOS(PingServoM, 0, PingServoR)
  waitcnt(clkfreq/2 + cnt)                      'Gives the PingServo moter time to move

    range := ping.Inches(bPING_Pin)             'Get Range In Inches
    'Debug.Str(String(13, "Is the Ping working?"))  '13 gives a charage return
    Debug.tx(Debug#CR)                              'Gives a charage return
    Debug.dec(range)                                'Gives the distance is inches via
    'waitcnt(clkfreq / 10 + cnt)                    'the Ping)))
    IF range => 23
      PingServoD := PingServoR
      Debug.Str(String("Hexapod is walking!"))
      bHexapodWalking                           'Goes to Pub "HexapodWalking" to start
    ELSE                                        'walking
      Debug.Str(String(13, "Left Blocked!"))
      PingServoR := PingServoR+75 '-75


PUB  fHexapodWalking
     Debug.Str(String(13, "Now we're in the fWalking part of the program!!!!!"))
     Debug.dec(PingServoD)
'*******************************************************************************************
'Converts the value/direction that the Ping))) servo is "looking" to the midpoint
'value for the FRHip
'*******************************************************************************************

     DirectionPW := PingServoD-350
     DirectionPW := fmath.fMul(DirectionPW, 0.375)
     DirectionPW := DirectionPW+600
     DirectionPW <#= 899                        'Max of 899
     DirectionPW #>= 601                        'Min of 601
     Debug.tx(Debug#CR)
     Debug.dec(DirectionPW)

'*******************************************************************************************
'Caculates the hip foot PW values and makes sure that they are in the proper range
```

```
'Caculates the hip foot PW values and makes sure that they are in the proper range
'*************************************************************************************

    Debug.Str(String(13,"DirectionPW before IF tree"))
    Debug.tx(Debug#CR)
    Debug.dec(DirectionPW)


    If DirectionPW == 750
      RPawHipPW  := DirectionPW - 150
      RPawFtSign := -1
      Debug.Str(String(13,"RPawHipPW == 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RPawHipPW)


      RFHipPW    := DirectionPW
      RFFtSign   := 1
      Debug.Str(String(13,"RFHipPW == 750, should be equal to 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RFHipPW)

      RRHipPW    := DirectionPW - 150
      RRFtSign   := -1
      Debug.Str(String(13,"RRHipPW == 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RRHipPW)

     '************************************
    ELSEIF DirectionPW =< 749
      RPawHipPW  := DirectionPW + 150
      RPawFtSign := -1
        IF RPawHipPW => 900
          RPawHipPW := RPawHipPW -300
          RPawFtSign := 1
        ELSEIF RPawHipPW =< 750
          RPawFtSign := 1

      Debug.Str(String(13,"RPawHipPW =< 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RPawHipPW)


      RFHipPW   := DirectionPW
      RFFtSign  := -1
        IF RFHipPW   => 751
          RFFtSign  := -1
      Debug.Str(String(13,"RFHipPW =< 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RFHipPW)
```

```
      Debug.dec(RFHipPW)

    RRHipPW    := DirectionPW + 150
    RRFtSign   := -1
      IF RRHipPW =< 600
        RRHipPW := RRHipPW +300
        RRFtSign := 1
      ELSEIF RPawHipPW =< 750
        RRFtSign := -1
      Debug.Str(String(13,"RRHipPW =< 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RRHipPW)

   ************************************
  ELSEIF DirectionPW => 750
    RPawHipPW  := DirectionPW - 150
    RPawFtSign := -1
      IF RPawHipPW =< 600
        RPawHipPW := RPawHipPW +300
        RPawFtSign := -1
      ELSEIF RPawHipPW =< 750
        RPawFtSign := 1
      Debug.Str(String(13,"RPawHipPW => 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RPawHipPW)

    RFHipPW   := DirectionPW
    RFFtSign  := -1
      IF RFHipPW   => 750
        RFFtSign  := 1
      Debug.Str(String(13,"RFHipPW => 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RFHipPW)

    RRHipPW    := DirectionPW - 150
    RRFtSign   := 1
      IF RRHipPW => 900
        RRHipPW := RRHipPW -300
        RRFtSign := 1
      ELSEIF RPawHipPW =< 750
        RRFtSign := 1
      Debug.Str(String(13,"RRHipPW =< 750"))
      Debug.tx(Debug#CR)
      Debug.dec(RRHipPW)

'*********************************************************************************
'Determins how much movement is in the hips and feet based on how close the hips
'position is to 750
'*********************************************************************************
RPawFootPW := ((RPawHipPW - 600)/2)-75       'Gives a value between -75 and 75
    Debug.Str(String(13,"RPawFootPW"))
```

```
RPawFootPW := ((RPawHipPW - 600)/2)-75        'Gives a value between -75 and 75
     Debug.Str(String(13,"RPawFootPW"))
     Debug.tx(Debug=CR)
     Debug.dec(RPawFootPW)

RFFootPW   := ((RFHipPW-600)/2)-75
     Debug.Str(String(13,"RFFootPW"))
     Debug.tx(Debug=CR)
     Debug.dec(RFFootPW)

RRFootPW   := ((RRHipPW-600)/2)-75
     Debug.Str(String(13,"RRFootPW"))
     Debug.tx(Debug=CR)
     Debug.dec(RRFootPW)

'*********************************************************************************************
'*********************************************************************************************
'Walking Sequence
'*********************************************************************************************
'LPaw, RF, LR Up
'***************
  PW := 900
  PSC.SETPOS(fLPawKnee, Ramp, PW)
  PSC.SETPOS(fRFKnee, Ramp, PW)
  PSC.SETPOS(fLRKnee, Ramp, PW)
  waitcnt(clkfreq / 8 + cnt)


'RPaw, LF, RR Backward; LPaw, RF, LR Forward
'*****************************************
'RPaw, LF, RR Backward
  'RPaw
  PW := RPawHipPW - (RPawFootPW* RPawFtSign)
  PSC.SETPOS(fRPawHip, Ramp, PW)
  PW := 762 - ((75 - (||RPawfootPW))* RPawFtsign)
  PSC.SETPOS(fRPawFoot, Ramp, PW)

'LF using opposite movement of RF
  PW := RFHipPW + (RFFootPW* RFFtSign)
  PSC.SETPOS(fLFHip, Ramp, PW)
  PW := 762 - (75 + ((RFFootPW*2)/2))
  PSC.SETPOS(fLFFoot, Ramp, PW)

'RR
  PW := RRHipPW - (RRFootPW* RRFtSign)
  PSC.SETPOS(fRRHip, Ramp, PW)
  PW := 762 - ((75 + (||RRFootPW))* RRFtSign)
  PSC.SETPOS(fRRFoot, Ramp, PW)

'**********************
'LPaw, RF, LR Forward
```

```
'************************
'LPaw, RF, LR Forward
'LPaw using opposite movement of RR
  PW := RRHipPW - (RRFootPW* RRFtSign)
  PSC.SETPOS(fLPawHip, Ramp, PW)
  PW := 762 - ((75+ (||RRFootPW)) * RRFtSign)
  PSC.SETPOS(fLPawFoot, Ramp, PW)

'RF
  PW := RFHipPW - (RFfootPW* RFFtSign)
  PSC.SETPOS(fRFHip, Ramp, PW)
  PW := 762 + (75- (||RFfootPW))
  PSC.SETPOS(fRFfoot, Ramp, PW)

'LR using opposite movement of RPaw
  PW := RPawHipPW - (RPawFootPW* RPawFtSign)
  PSC.SETPOS(fLRHip, Ramp, PW)
  PW := 762 - ((75- (||RPawFootPW))*RPawFtSign)
  PSC.SETPOS(fLRFoot, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)


'LPaw, RF, LR Down
'*****************
  PW := 749
  PSC.SETPOS(fLPawKnee, Ramp, PW)
  PSC.SETPOS(fRFKnee, Ramp, PW)
  PSC.SETPOS(fLRKnee, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

'RPaw, LF, RR Up
'***************
  PW := 900
  PSC.SETPOS(fRPawKnee, Ramp, PW)
  PSC.SETPOS(fLFKnee, Ramp, PW)
  PSC.SETPOS(fRRKnee, Ramp, PW)
  waitcnt(clkfreq / 8 + cnt)


'RPaw, LF, RR Forward; LPaw, RF, LR Backward
'*******************************************
'RPaw, LF, RR Forward
'RPaw
  PW := RPawHipPW - (RPawFootPW* RPawFtSign)
  PSC.SETPOS(fRPawHip, Ramp, PW)
  PW := 762 + ((75 - (||RPawFootPW)) * RPawFtSign)
  PSC.SETPOS(fRPawFoot, Ramp, PW)

'LF using opposite movement of RF
  PW := RFHipPW - (RFfootPW* RFFtSign)
  PSC.SETPOS(fLFHip, Ramp, PW)
```

```
  PW := RFHipPW - (RFfootPW* RFFtSign)
  PSC.SETPOS(fLFHip, Ramp, PW)
  PW := 762 + (75 + (||RFFootPW))
  PSC.SETPOS(fLFFoot, Ramp, PW)

'RR
  PW := RRHipPW + (RRFootPW* RRFtSign)
  PSC.SETPOS(fRRHip, Ramp, PW)
  PW := 762 + ((75 + (||RRFootPW))* RRFtSign)
  PSC.SETPOS(fRRFoot, Ramp, PW)

'**********************
'LPaw, RF, LR Backwards
'LPaw using opposite movement of RR
  PW := RRHipPW - (RRFootPW* RRFtSign)
  PSC.SETPOS(fLPawHip, Ramp, PW)
  PW := 762 - ((75 - (||RRFootPW))* RRFtSign)
  PSC.SETPOS(fLPawFoot, Ramp, PW)

'RF
  PW := RFHipPW - (RFFootPW* RFFtSign)
  PSC.SETPOS(fRFHip, Ramp, PW)
  PW := 762 + (75 - (||RFFootPW))
  PSC.SETPOS(fRFFoot, Ramp, PW)

'LR using opposite movement of RPaw
  PW := RPawHipPW + (RPawFootPW* RPawFtSign)
  PSC.SETPOS(fLRHip, Ramp, PW)
  PW := 762 - ((75 - (||RPawFootPW))* RPawFtSign)
  PSC.SETPOS(fLRFoot, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

'RPaw, LF, RR Down
'******************
  PW := 749
  PSC.SETPOS(fRPawKnee, Ramp, PW)
  PSC.SETPOS(fLFKnee, Ramp, PW)
  PSC.SETPOS(fRRKnee, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

  MainProgramf                          'Send the program back to "MainProgramf"



'*****************************************************************************
'*****************************************************************************
'*****************************************************************************
'*****************************************************************************
'*****************************************************************************
'*****************************************************************************
'*****************************************************************************
```

```
'*****************************************************************************


PUB  bHexapodWalking
     Debug.Str(String(13, "Now we're in the bbbWalking part of the program!!!!!"))
     Debug.dec(PingServoD)
'*****************************************************************************
'Converts the value/direction that the Ping))) servo is "looking" to the midpoint
' value for the FRHip
'*****************************************************************************
     DirectionPW := PingServoD-350
     DirectionPW := fmath.fMul(DirectionPW, 0.375)
     DirectionPW := DirectionPW+600
     DirectionPW <#= 899                  'Max of 899
     DirectionPW #>= 601                  'Min of 601
     Debug.tx(Debug#CR)
     Debug.dec(DirectionPW)
     'Debug.tx(Debug#CR)
     'Debug.str(fstring.floatToString(DirectionPW))   'This is turned off right now
     'PSC.SETPOS(RFHip, Ramp, RFHipPW)

'*****************************************************************************
'Caculates the hip foot PW values and makes sure that they are in the proper range
'*****************************************************************************
     'DirectionPW := 752                          'for debugging
     Debug.Str(String(13,"DirectionPW before IF tree"))
     Debug.tx(Debug#CR)
     Debug.dec(DirectionPW)


     If DirectionPW == 750
       RPawHipPW  := DirectionPW - 150
       RPawFtSign := -1
       Debug.Str(String(13,"RPawHipPW == 750"))
       Debug.tx(Debug#CR)
       Debug.dec(RPawHipPW)


       RFHipPW    := DirectionPW
       RFFtSign   := -1
       Debug.Str(String(13,"RFHipPW == 750, should be equal to 750"))
       Debug.tx(Debug#CR)
       Debug.dec(RFHipPW)

       RRHipPW    := DirectionPW - 150
       RRFtSign   := 1
       Debug.Str(String(13,"RRHipPW == 750"))
       Debug.tx(Debug#CR)
       Debug.dec(RRHipPW)
```

```
    Debug.dec(RRHipPW)

 ************************************
ELSEIF DirectionPW =< 749
    RPawHipPW  := DirectionPW + 150
    RPawFtSign := -1
      IF RPawHipPW => 900
        RPawHipPW := RPawHipPW -300
        RPawFtSign := -1
      ELSEIF RPawHipPW =< 750
        RPawFtSign := -1

    Debug.Str(String(13,"RPawHipPW =< 750"))
    Debug.tx(Debug#CR)
    Debug.dec(RPawHipPW)


    RFHipPW    := DirectionPW
    RFFtSign   := 1
      IF RFHipPW    => 751
        RFFtSign   := -1
    Debug.Str(String(13,"RFHipPW =< 750"))
    Debug.tx(Debug#CR)
    Debug.dec(RFHipPW)

    RRHipPW    := DirectionPW - 150
    RRFtSign   := 1
      IF RRHipPW =< 600
        RRHipPW := RRHipPW +300
        RRFtSign := -1
      ELSEIF RPawHipPW =< 750
        RRFtSign := -1
    Debug.Str(String(13,"RRHipPW =< 750"))
    Debug.tx(Debug#CR)
    Debug.dec(RRHipPW)

 ************************************
ELSEIF DirectionPW => 750
    RPawHipPW  := DirectionPW + 150
    RPawFtSign := -1
      IF RPawHipPW =< 600
        RPawHipPW := RPawHipPW +300
        RPawFtSign := 1
      ELSEIF RPawHipPW =< 750
        RPawFtSign := 1
    Debug.Str(String(13,"RPawHipPW => 750"))
    Debug.tx(Debug#CR)
    Debug.dec(RPawHipPW)

    RFHipPW    := DirectionPW
```

```
      RFHipPW    := DirectionPW
      RFFtSign   := 1
        IF RFHipPW    => 750
          RFFtSign   := 1
        Debug.Str(String(13,"RFHipPW => 750"))
        Debug.tx(Debug#CR)
        Debug.dec(RFHipPW)

      RRHipPW    := DirectionPW + 150
      RRFtSign   := -1
        IF RRHipPW => 900
          RRHipPW := RRHipPW +300
          RRFtSign := 1
        ELSEIF RPawHipPW =< 750
          RRFtSign := -1
        Debug.Str(String(13,"RRHipPW =< 750"))
        Debug.tx(Debug#CR)
        Debug.dec(RRHipPW)

'*****************************************************************************
'Determins how much movement is in the hips and feet based on how close the hips
'position is to 750
'*****************************************************************************
RPawFootPW := ((RPawHipPW - 600)/2)-75            'Gives a value between -75 and 75
     Debug.Str(String(13,"RPawFootPW"))
     Debug.tx(Debug#CR)
     Debug.dec(RPawFootPW)

RFFootPW   := ((RFHipPW-600)/2)-75
     Debug.Str(String(13,"RFFootPW"))
     Debug.tx(Debug#CR)
     Debug.dec(RFFootPW)

RRFootPW   := ((RRHipPW-600)/2)-75
     Debug.Str(String(13,"RRFootPW"))
     Debug.tx(Debug#CR)
     Debug.dec(RRFootPW)

'*****************************************************************************
'*****************************************************************************
'Walking Sequence
'*****************************************************************************

'LPaw, RF, LR Up
'***************
  PW := 900
  PSC.SETPOS(bLPawKnee, Ramp, PW)
  PSC.SETPOS(bRFKnee, Ramp, PW)
  PSC.SETPOS(bLRKnee, Ramp, PW)
  waitcnt(clkfreq / 8 + cnt)
```

```
'RPaw, LF, RR Backward; LPaw, RF, LR Forward
'****************************************
'RPaw, LF, RR Backward
  'RPaw
  PW := RPawHipPW + (RPawFootPW* RPawFtSign)
  PSC.SETPOS(bRPawHip, Ramp, PW)
  PW := 762 + ((75 + (||RPawfootPW))* RPawFtsign)
  PSC.SETPOS(bRPawFoot, Ramp, PW)

'LF using opposite movement of RF
  PW := RFHipPW + (RFFootPW* RFFtSign)
  PSC.SETPOS(bLFHip, Ramp, PW)
  PW := 762 + (75 + ((RFFootPW*2)/2))
  PSC.SETPOS(bLFFoot, Ramp, PW)

'RR
  PW := RRHipPW + (RRFootPW* RRFtSign)
  PSC.SETPOS(bRRHip, Ramp, PW)
  PW := 762 + ((75 + (||RRFootPW))* RRFtSign)
  PSC.SETPOS(bRRFoot, Ramp, PW)

'************************
'LPaw, RF, LR Forward
'LPaw using opposite movement of RR
  PW := RRHipPW - (RRFootPW* RRFtSign)
  PSC.SETPOS(bLPawHip, Ramp, PW)
  PW := 762 - ((75+ (||RRFootPW))* RRFtSign)
  PSC.SETPOS(bLPawFoot, Ramp, PW)

'RF
  PW := RFHipPW - (RFfootPW* RFFtSign)
  PSC.SETPOS(bRFHip, Ramp, PW)
  PW := 762 - (75+ (||RFfootPW))
  PSC.SETPOS(bRFfoot, Ramp, PW)

'LR using opposite movement of RPaw
  PW := RPawHipPW - (RPawFootPW* RPawFtSign)
  PSC.SETPOS(bLRHip, Ramp, PW)
  PW := 762 - ((75+ (||RPawFootPW))*RPawFtSign)
  PSC.SETPOS(bLRFoot, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)


'LPaw, RF, LR Down
'*****************
  PW := 749
  PSC.SETPOS(bLPawKnee, Ramp, PW)
  PSC.SETPOS(bRFKnee, Ramp, PW)
  PSC.SETPOS(bLRKnee, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)
```

```
  PSC.SETPOS(bLRKnee, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

'RPaw, LF, RR Up
'***************
  PW := 900
  PSC.SETPOS(bRPawKnee, Ramp, PW)
  PSC.SETPOS(bLFKnee, Ramp, PW)
  PSC.SETPOS(bRRKnee, Ramp, PW)
  waitcnt(clkfreq / 8 + cnt)


'RPaw, LF, RR Forward; LPaw, RF, LR Backward
'*****************************************
'RPaw, LF, RR Forward
'RPaw
  PW := RPawHipPW - (RPawFootPW* RPawFtSign)
  PSC.SETPOS(bRPawHip, Ramp, PW)
  PW := 762 + ((75 - (RPawFootPW)) * RPawFtSign)
  PSC.SETPOS(bRPawFoot, Ramp, PW)

'LF using opposite movement of RF
  PW := RFHipPW + (RFfootPW* RFFtSign)
  PSC.SETPOS(bLFHip, Ramp, PW)
  PW := 762 - (75 + (RFFootPW))
  PSC.SETPOS(bLFFoot, Ramp, PW)

'RR
  PW := RRHipPW + (RRFootPW* RRFtSign)
  PSC.SETPOS(bRRHip, Ramp, PW)
  PW := 762 - ((75 + (RRFootPW)) * RRFtSign)
  PSC.SETPOS(bRRFoot, Ramp, PW)

'**********************
'LPaw, RF, LR Backwards
'LPaw using opposite movement of RR
  PW := RRHipPW + (RRFootPW* RRFtSign)
  PSC.SETPOS(bLPawHip, Ramp, PW)
  PW := 762 - ((75 - (RRFootPW)) + RRFtSign)
  PSC.SETPOS(bLPawFoot, Ramp, PW)

'RF
  PW := RFHipPW + (RFFootPW* RFFtSign)
  PSC.SETPOS(bRFHip, Ramp, PW)
  PW := 762 + (75 - (RFFootPW))
  PSC.SETPOS(bRFFoot, Ramp, PW)

'LR using opposite movement of RPaw
  PW := RPawHipPW + (RPawFootPW+ RPawFtSign)
  PSC.SETPOS(bLRHip, Ramp, PW)
  PW := 762 + ((75 + (RPawFootPW)) * RPawFtSign)
```

```
  PW := RFHipPW + (RFFootPW* RFFtSign)
  PSC.SETPOS(bRFHip, Ramp, PW)
  PW := 762 + (75 - (RFFootPW))
  PSC.SETPOS(bRFFoot, Ramp, PW)

'LR using opposite movement of RPaw
  PW := RPawHipPW + (RPawFootPW* RPawFtSign)
  PSC.SETPOS(bLRHip, Ramp, PW)
  PW := 762 + ((75 + (RPawFootPW))* RPawFtSign)
  PSC.SETPOS(bLRFoot, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

'RPaw, LF, RR Down
'*****************
  PW := 749
  PSC.SETPOS(bRPawKnee, Ramp, PW)
  PSC.SETPOS(bLFKnee, Ramp, PW)
  PSC.SETPOS(bRRKnee, Ramp, PW)
  waitcnt(clkfreq / 6 + cnt)

  MainProgramB                    'Send the program back to "MainProgramB"
```

Acknowledgments

I thank my wife for her love and support in everything