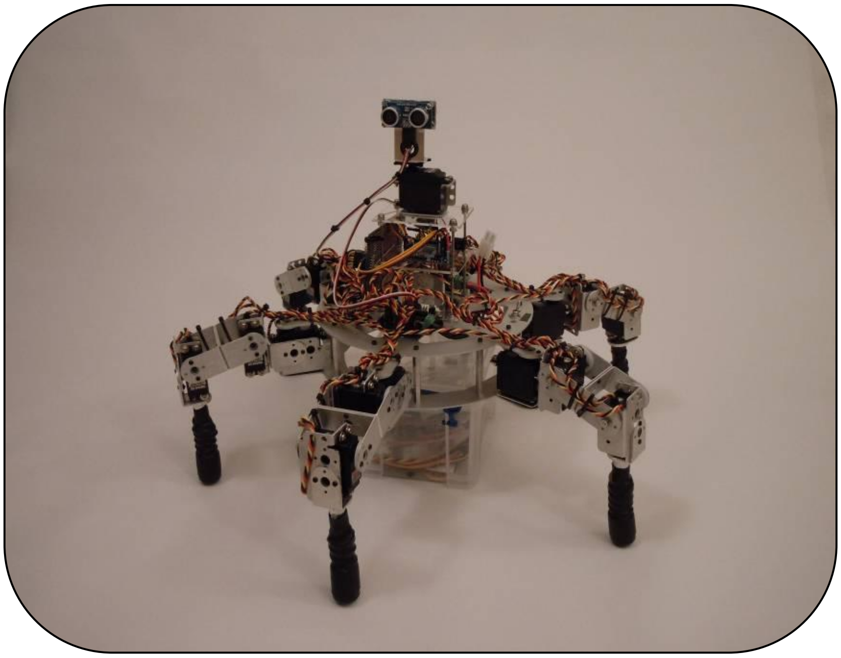
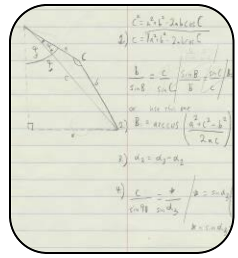
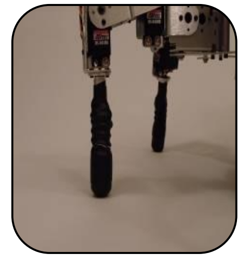
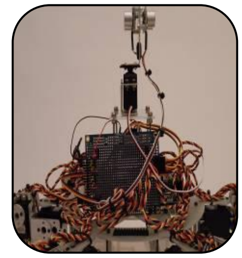






HexWalker ©

Terrain Adaptive, Omnidirectional Hexapod Walker

David Dorhout

vanmunch@yahoo.com

YouTube: vanmunch36

December 31st, 2010

SchmartBoard Propeller Design Contest

Project Number: PO006

Table of Contents

Project Number

Project Number: PO006

Project Description

Purpose

The purpose of this unnamed robot is to develop an adaptive terrain program for Prospero, my robotic farming robot. Currently, Prospero is using a walking program that I originally developed for Parallax's Basic Stamp2sx (BS2sx). That program allows Prospero to autonomously avoid obstacles and instantly change directions without turning its body. However, that program had to fit inside and use the limited variable space inside the BS2sx. On the other hand, Prospero uses Parallax's powerful Propeller chip that along with 64K of global RAM/ROM and 40 I/O pins has eight 32-bit processors that allow for true multi-processing. All of this gives me the ability to create a robot that is capable of dynamically adapting to its terrain and walk over radically uneven surfaces.

Electronics and Hardware

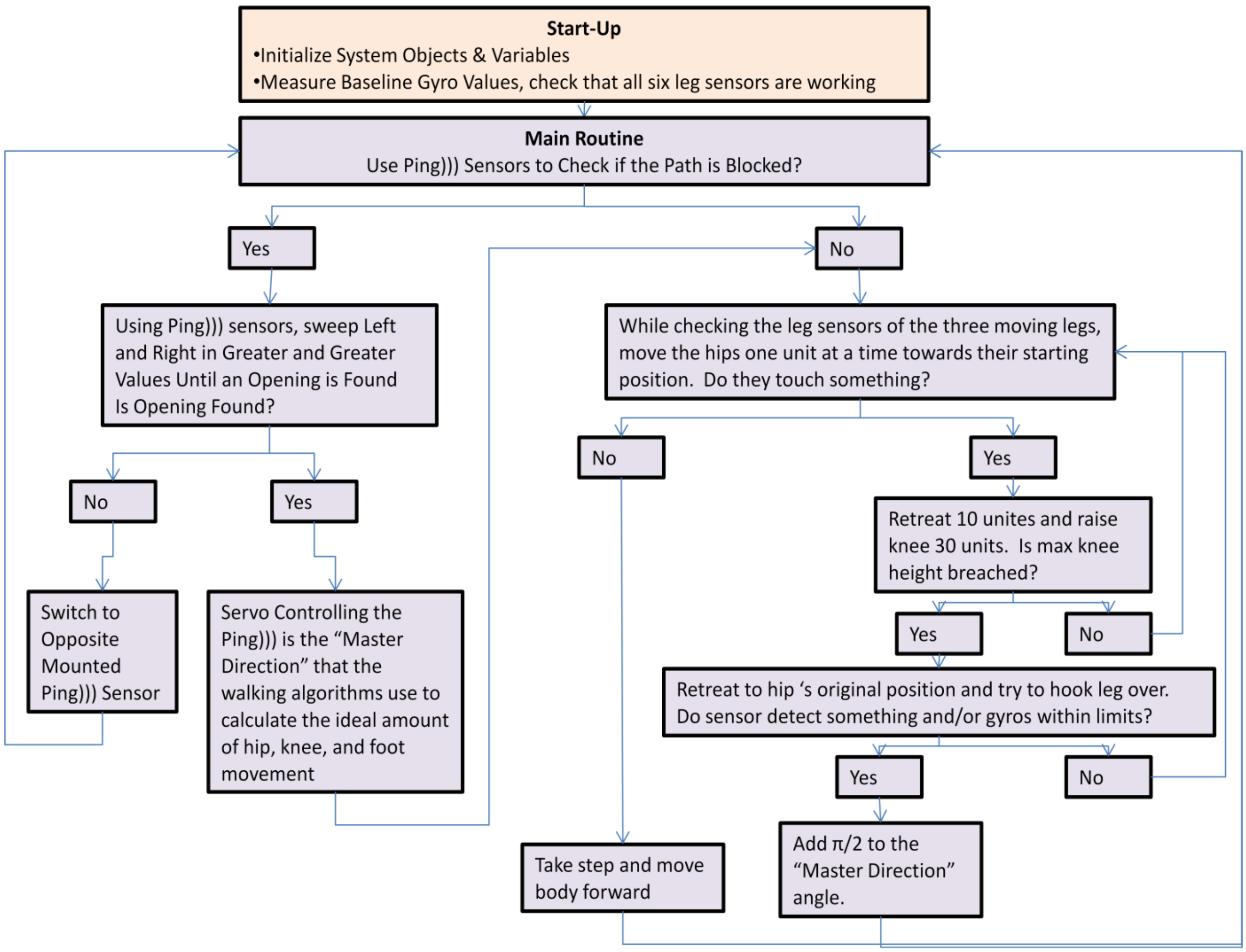
Parallax's Propeller chip is seated inside the handy Schmart Board that allows for easy access to all of the pins for rapid prototyping. Its inexpensive price point and modular form also allows for accidental shorts that sometimes happen while programming something with this many servos and response-based behavior algorithms (Not that I have direct experience with this :)).

For general obstacle avoidance the robot has opposite mounted Ping))) sensors mounted on a 180° servo giving a full 360° view. In the center of the body are two LISY300 gyros mounted in the vertical and horizontal planes. This allows the robot to stay relatively parallel to the ground or know if the slope is too steep or the obstacle is too tall. Finally, each leg has a "toe" sensor that covers the bottom half of the leg. It's constructed by floating a metal spring just over the surface of the aluminum tube that serves as the rigid support for the leg. That metal tube is insulated from the rest of the robot's body and energized with a small amount of current. The circuit is complete once the metal spring surrounding the leg tube bumps into something on the side or if it comes into contact with the ground. The legs' movements are broken into first the XY movement and then into Z movement. This allows the same sensor to distinguish between contact with a vertical obstacle and contact with the ground.

Behavior

This robot uses a 3 by 3 leg gait where three legs are always on the ground with three in the air moving. The robot walks in the direction that the "head" with the Ping))) sensors is looking. This is the "Master Direction." The program then modifies the angle of the master direction for each of the six legs based on their relative position. The program then calculates the amount of hip and foot movement that is required for that leg unit to produce a vector in alignment with the master direction. The magnitude of the vector is determined by the speed of the robot that is determined by the terrain. Once the three legs know their hip starting positions they lift themselves up and attempt to make it to that position one unit of movement at a time, checking to see if the tow sensor has hit something. If it has, it retreats a little, and lifts itself higher and tries again. If it continues to hit something and all of the safety limits have been reached it adds 90° to the master angle and tries for that direction.

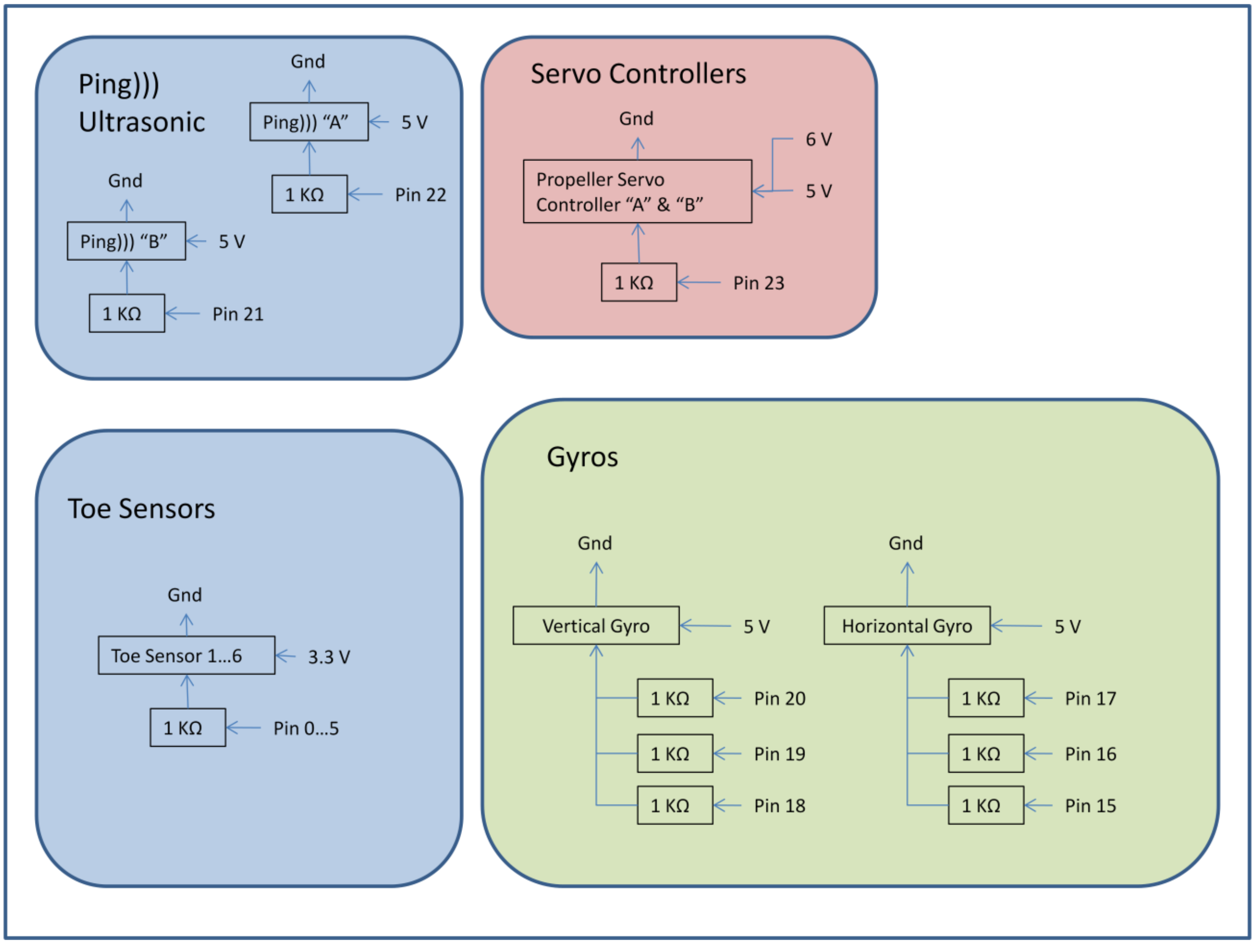
Block Diagram



Bill Of Materials

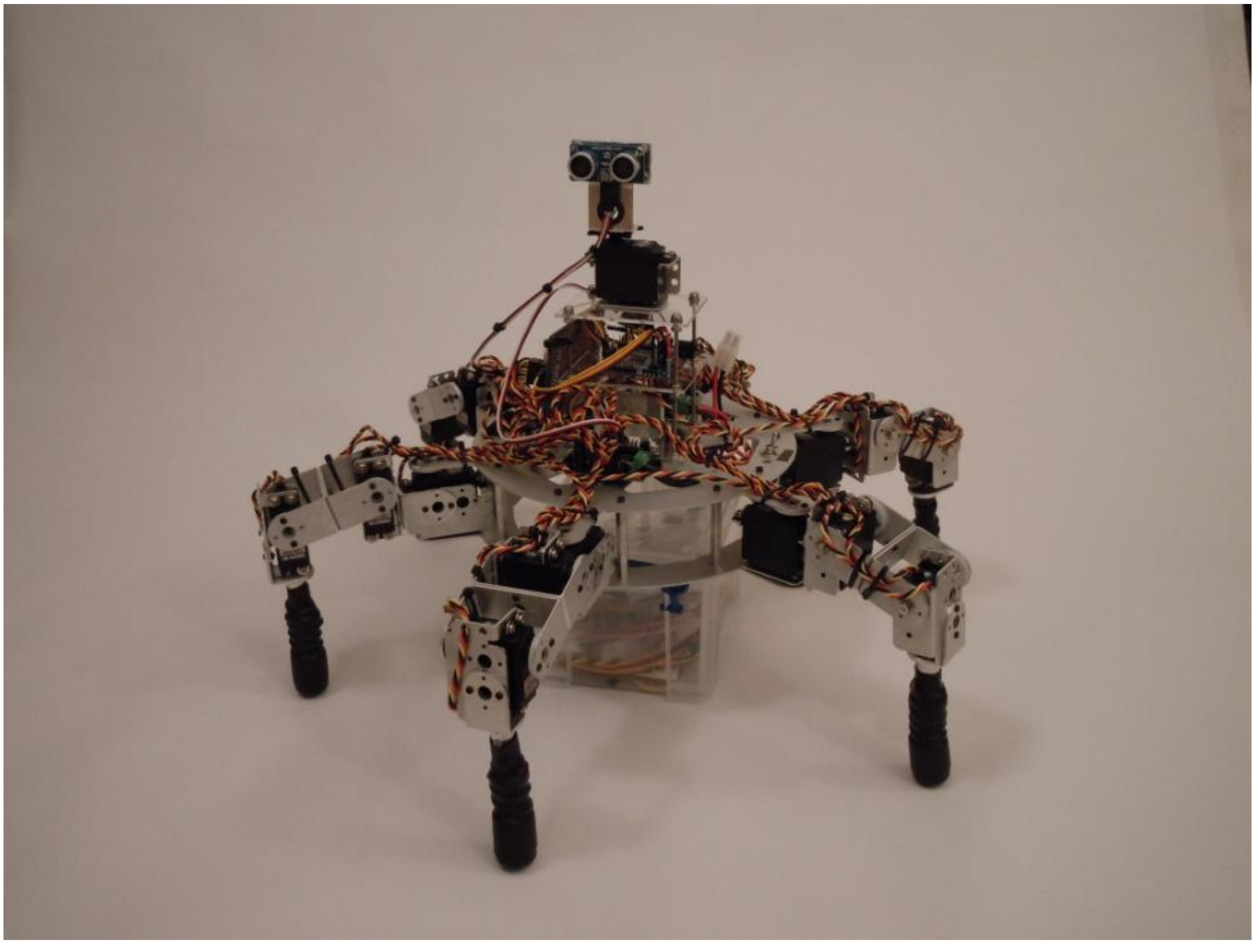
Qty	Description	Company	Part Number
1	AH3-R (no electronics; no servos)	Lynxmotion	AH3RCA
1	6.0 V Ni-MH 2800 mAh Battery	Lynxmotion	BAT-05
18	HS-645 Servo	Tower Hobby	LM3122
8	Extender Cable- 6"	Lynxmotion	SEA-01
1	Aluminum Multi-Purpose Servo Bracket	Lynxmotion	ASB-04
1	Parallax Propeller SchmartModule	Schmart Board	710-0005-01
2	Propeller Servo Controller USB	Parallax	28830
2	LISY300 Gyroscope Module	Parallax	27922
1	PING))) Ultrasonic Sensor with Mounting Bracket	Parallax	910-28015A
1	PING))) Ultrasonic Sensor	Parallax	28015
1	Parallax Blank 3x4 Proto Board	Parallax	45305
6	Resistors for legs 10K ohm,1/4 Watt Resistor	Parallax	150-01030
4	100 ohm Resistor, 1/4 Watt	Parallax	150-01011
1	Solderless Breadboard	Parallax	700-00012
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
	Aluminum Tubing, Sheeting and Rods	Various	-
	1/8" Plexiglas	Various	-
	3/4" Springs	Various	-
	Heat Shrink Tubing	Various	-
	3/4" wooden dowel	Various	-

Schematic



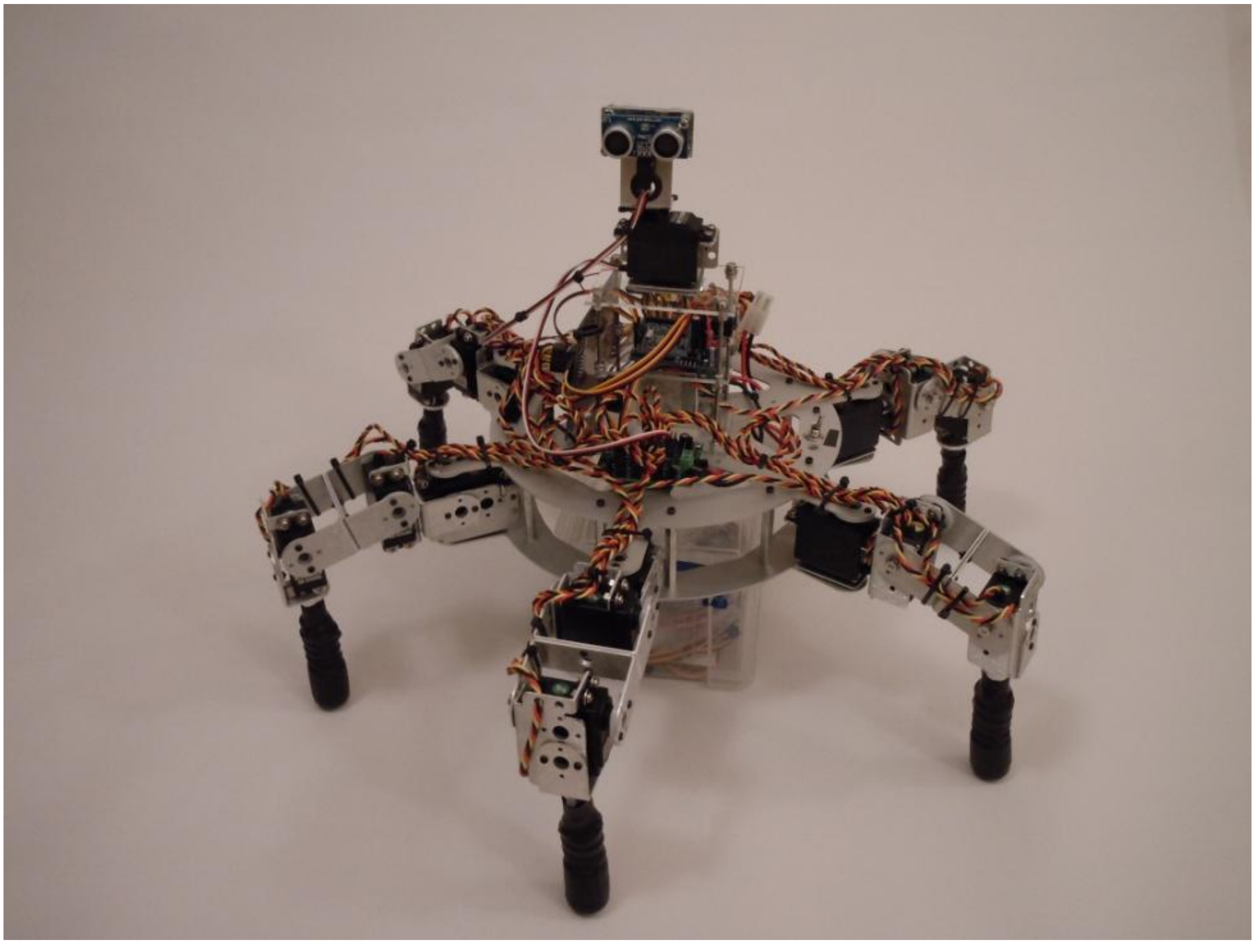
Ping)))
Ultrasonic
Gnd
Ping))) "A"
5 V
1 KΩ
Pin 22
Gnd
Ping))) "B"
5 V
1 KΩ
Pin 21
Servo Controllers
Gnd
6 V
Propeller Servo
Controller "A" & "B"
5 V
1 KΩ
Pin 23
Toe Sensors
Gnd
Toe Sensor 1…6
3.3 V
1 KΩ
Pin 0…5
Gyros
Gnd
Vertical Gyro
5 V
1 KΩ
Pin 20
1 KΩ
Pin 19
1 KΩ
Pin 18
Gnd
Horizontal Gyro
5 V
1 KΩ
Pin 17
1 KΩ
Pin 16
1 KΩ
Pin 15

Photographs

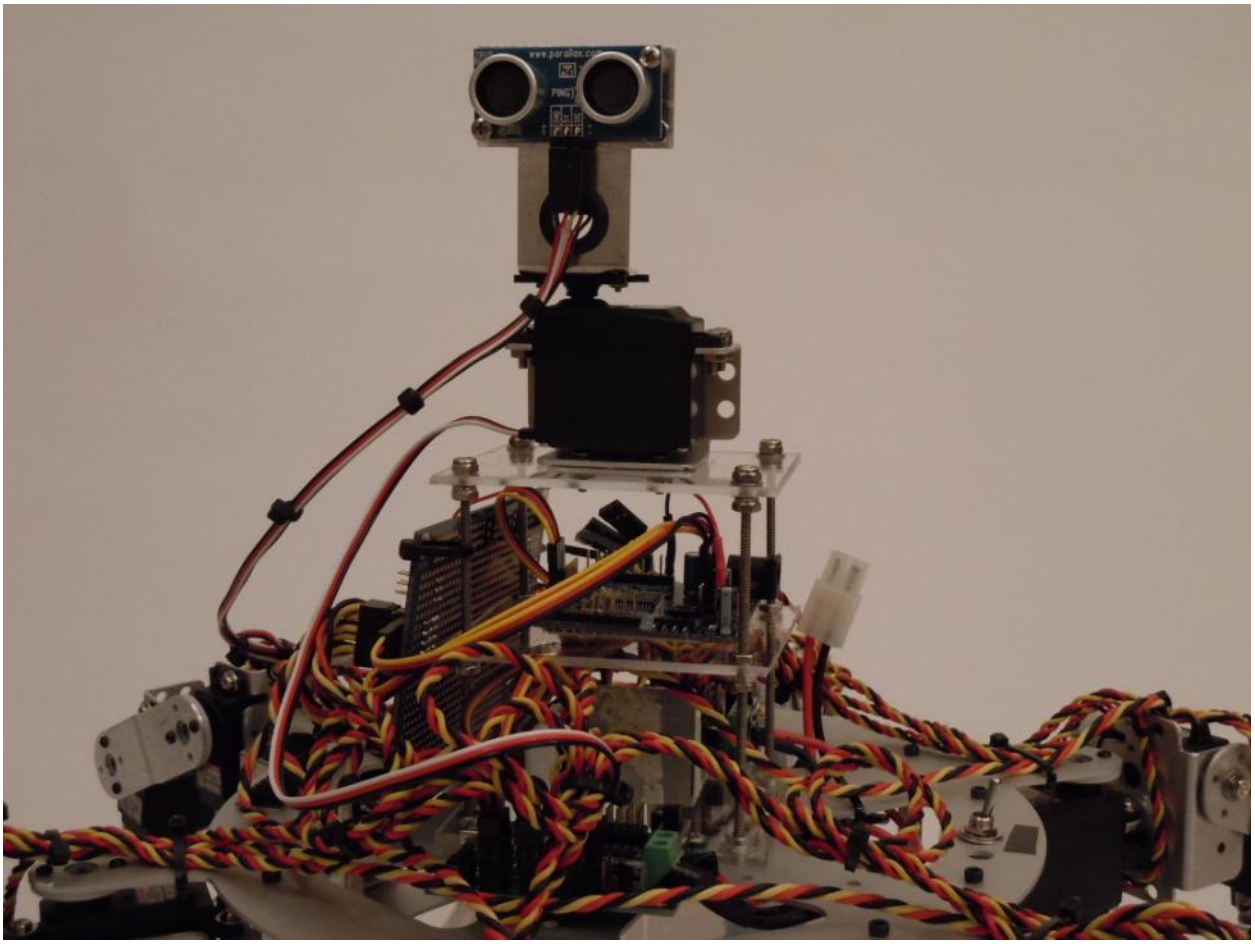


"Front*" view

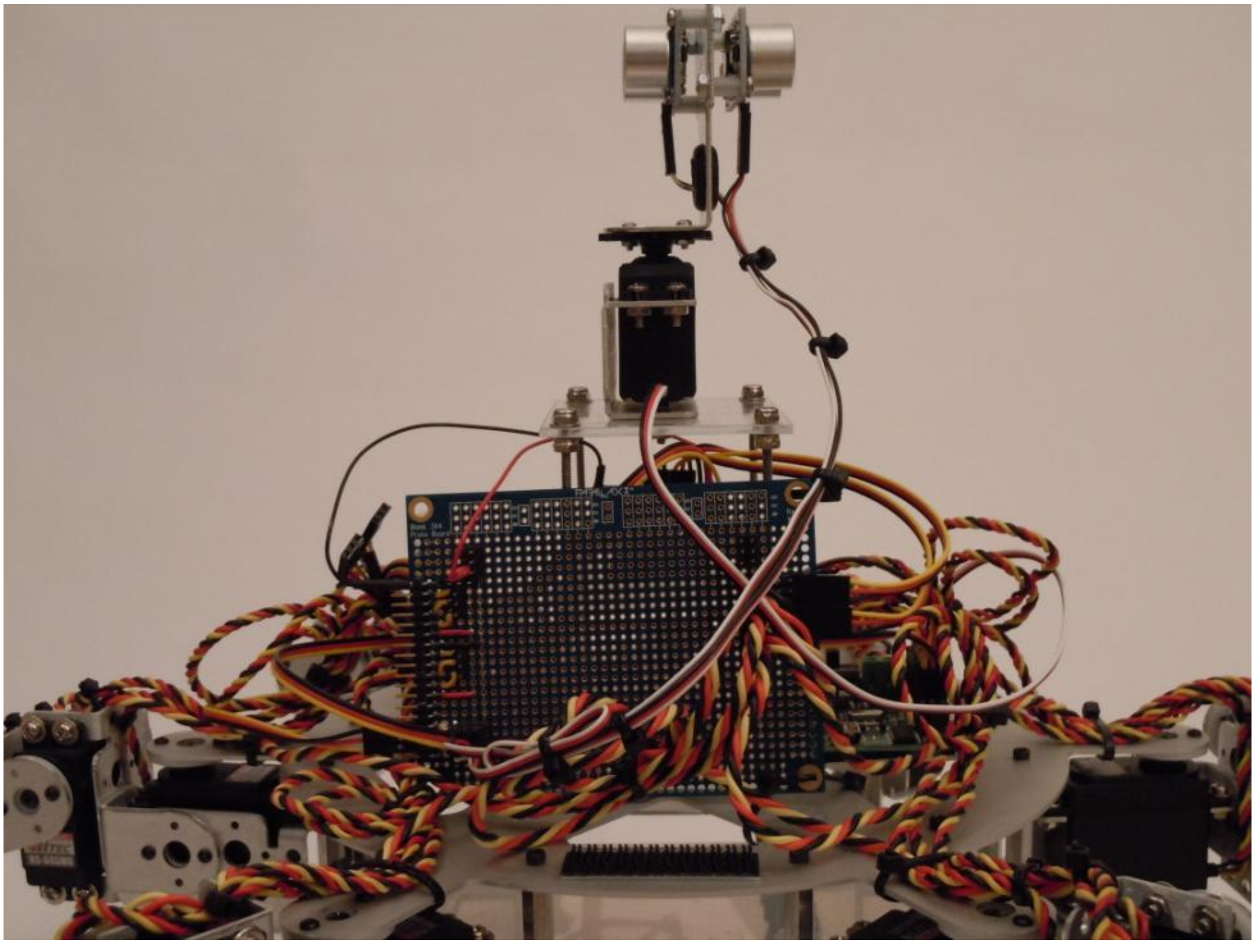
* the robot is functionally symmetrical so it doesn't have true "sides"



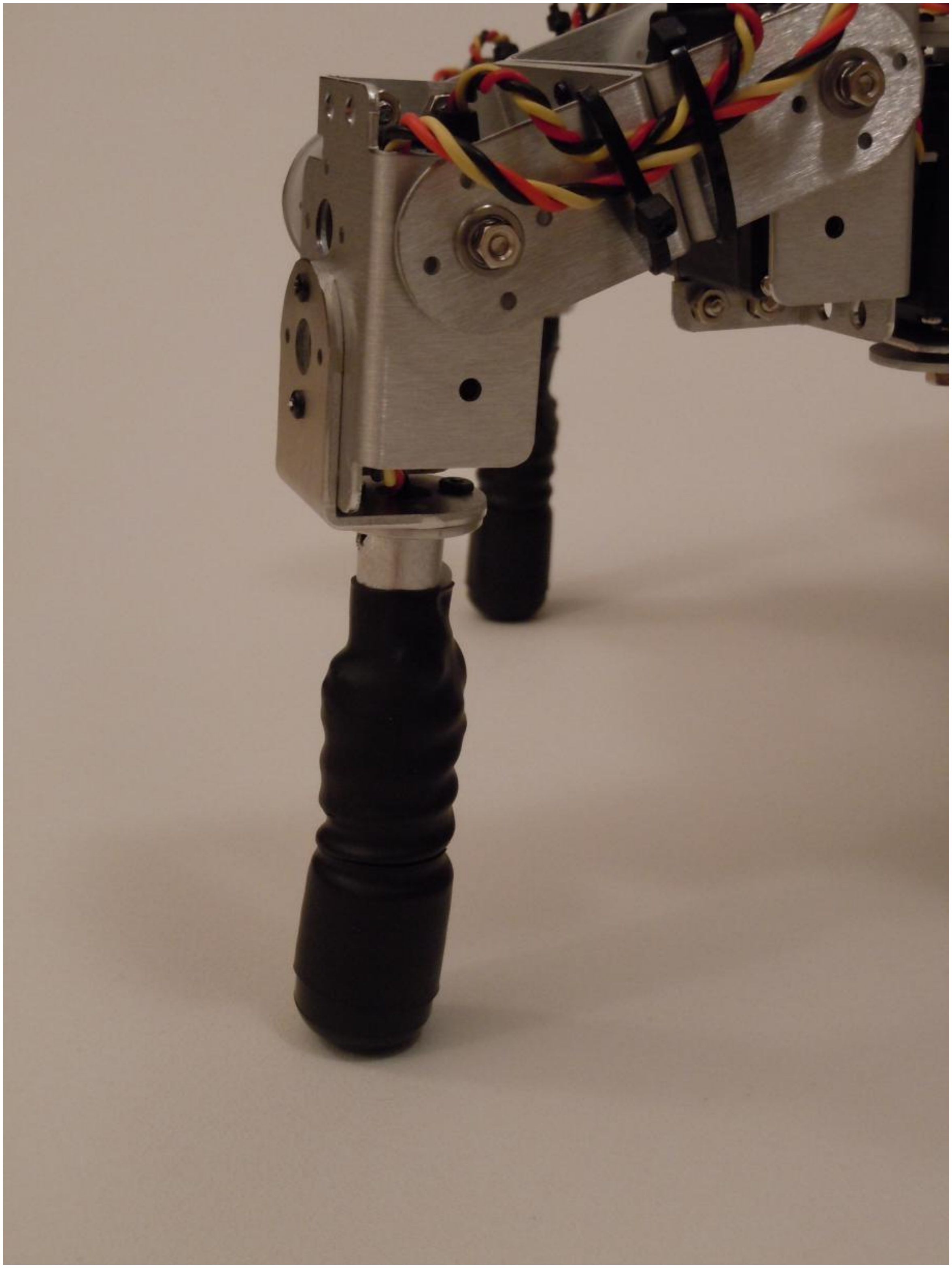
Overhead view



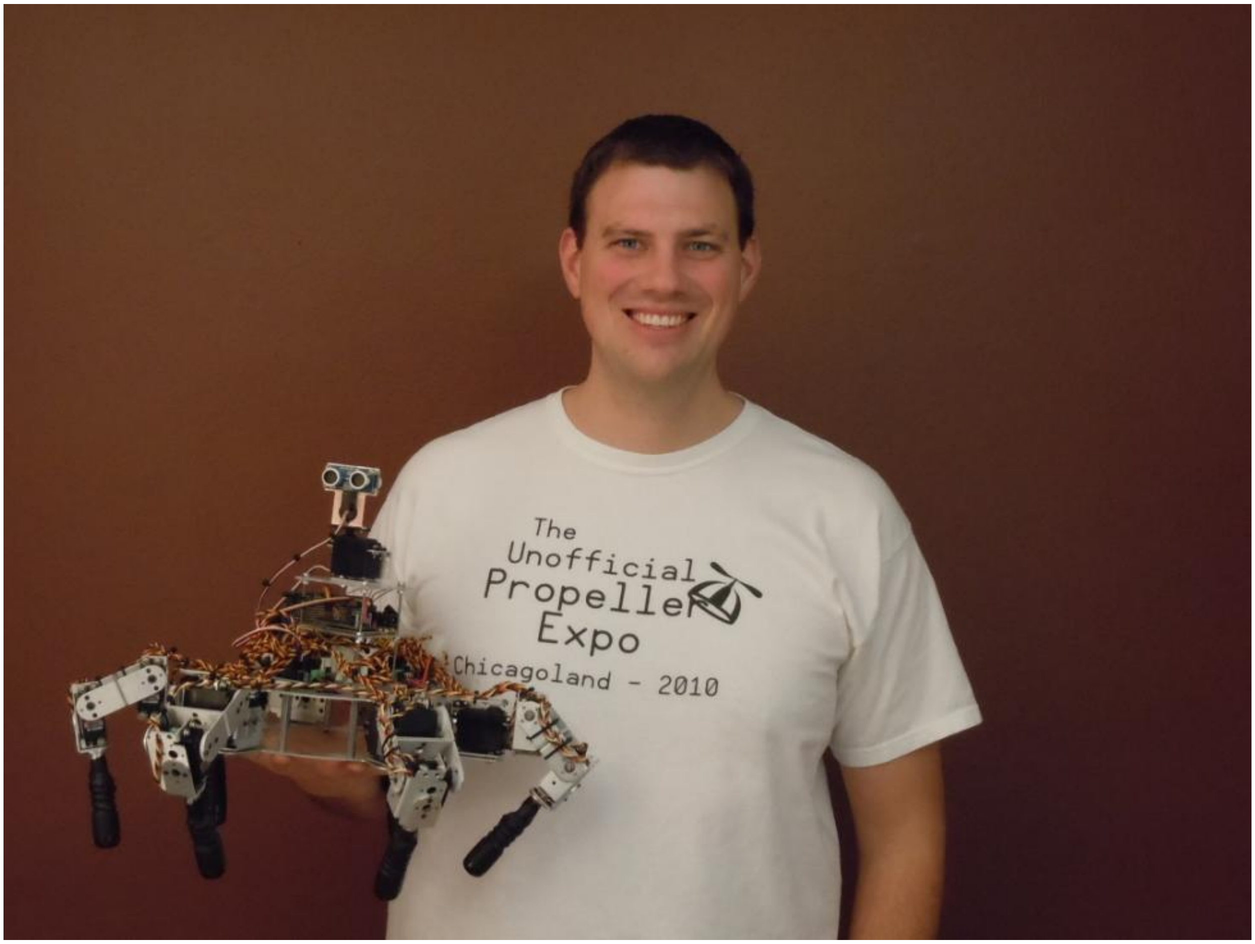
Close up of Propeller Schmart Board and gyros in the bottom center



Parallax prototyping board



Close-up of one of the toe sensors covered in heat shrink.



The Author and builder with the robot

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.

The code is not yet complete so below are the different modules of the code that I have been working on as well as some of my notes. Before the board burned out it was able to:

- Detect objects with the Ping))) sensors
- Detect objects with its toe sensors and move the leg around them in order to get to the legs starting point.
- Use the gyros to know the plane orientation of the robot
- Calculate the leg's location relative to the body for determining how much of the hip's moment needed to be adjusted based on the overall legs position.

```
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             = 0                 'Ramp is the speed between 0-63 that the PSC turns the servos  (fast to slow)



'Constants for the leg servos
RPawFoot        = 31
RPawKnee        = 30
RPawHip         = 29

RFFoot          = 26
RFKnee          = 27
RFHip           = 28

RRFoot          = 23
RRKnee          = 22
RRHip           = 21

LPawFoot        =  0
LPawKnee        =  1
LPawHip         =  2

LFFoot          = 15
LFKnee          = 14
LFHip           = 13

LRFoot          =  3
LRKnee          =  4
LRHip           =  5

'The maximune height that the knees can go up
KneeHightMax    =  1000

'The maximune height that the three down legs can go in an attempt to raise
'the body over an obstical
BodyHeightMax   =  510

'Toe Sensor Pins
RPawToeSensor   =  0
RFToeSensor     =  1
RRToeSensor     =  2
LPawToeSensor   =  3
```

```
'Toe Sensor Pins
RPawToeSensor   =  0
RFToeSensor     =  1
RRToeSensor     =  2
LPawToeSensor   =  3
LFToeSensor     =  4
LRToeSensor     =  5


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

'Variables for the PW of each leg servo
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    Problem1
'long    Problem2
'long    Problem3
```

```
'long    Problem1
'long    Problem2
'long    Problem3
'long    CogNumber

'long  stack[200]                      'Cog stack space
'byte  cog                             'Cog ID

'long    leg


'Variables for Leg Placement
long                    toeX
long                    toeY
long                    toeZ

Long                    HipX
Long                    HipY
Long                    HipZ

Long                    KneeX
Long                    KneeY
Long                    KneeZ

Long                    FootX
Long                    FootY
Long                    FootZ

long                    toeA
long                    toeB
long                    toeC

Long                    HipA
Long                    HipB
Long                    HipC

Long                    KneeA
Long                    KneeB
Long                    KneeC

Long                    FootA
Long                    FootB
Long                    FootC


Long                    KneeXPW
Long                    KneeYPW
Long                    KneeZPW

Long                    LegStateX
```

```
Long                    KneeZPW

Long                    LegStateX
Long                    LegStateY
Long                    LegStateZ
Long                    LegStateXYZ

Long                    DirectionX
Long                    DirectionY
Long                    DirectionZ

Long                    StartX
Long                    StartY
Long                    StartZ

Long                    FinishX
Long                    FinishY
Long                    FinishZ

Long                    ToeSensorX
Long                    ToeSensorY
Long                    ToeSensorZ

Long                    HipXPlacement
Long                    HipYPlacement
Long                    HipZPlacement


Long                    BodyHeight
Long                    StuckLegHip
Long                    StuckLegHipStart
Long                    StuckLegHipFinish
Long                    StuckLegKnee
Long                    StuckLegFoot
Long                    StuckLegToeSensor
Long                    StuckLegDirection

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

PUB StartUp
''Starts up and Initializes leg values from 0 to 750
    waitcnt(clkfreq*5 + cnt)
```

```
''Starts up and Initializes leg values from 0 to 750
    waitcnt(clkfreq*5 + cnt)

    PSC.START(COMPIN, PSC_BAUD)
    Debug.Start(31, 30, 0, 57600)
    Debug.tx(Debug#CLS)
    Debug.str(string(13, "Starting! :) "))
    Debug.str(string(13, "Initializing leg values! :) "))

    RPawFootPW      := 750
    RPawKneePW      := 750
    RPawHipPW       := 750

    RFFootPW        := 750
    RFKneePW        := 750
    RFHipPW         := 750

    RRFootPW        := 750
    RRKneePW        := 750
    RRHipPW         := 750

    LPawFootPW      := 750
    LPawKneePW      := 750
    LPawHipPW       := 750

    LFFootPW        := 750
    LFKneePW        := 750
    LFHipPW         := 750

    LRFootPW        := 750
    LRKneePW        := 750
    LRHipPW         := 750

    HipX            := 750
    HipY            := 750
    HipZ            := 750

    KneeX           := 750
    KneeY           := 750
    KneeZ           := 750

    KneeXPW         := 750
    KneeYPW         := 750
    KneeZPW         := 750

    HipXPlacement   := 750
    HipYPlacement   := 750
    HipZPlacement   := 750

    BodyHeight      := 750
```

```
  BodyHeight     := 750

  waitcnt(clkfreq*1 + cnt)
  LegSetUp

PUB LegSetUp
  Debug.str(string(13, "LegSetUp! :) "))

  StartX  := 750
  FinishX := 600
  HipXPlacement := StartX

  StartY  := 750
  FinishY := 750
  HipYPlacement := StartY

  StartZ  := 750
  FinishZ := 950
  HipZPlacement := StartZ

  HipX  := RPawHip
  KneeX := RPawKnee
  FootX := RPawFoot
  toeX  := RPawToeSensor

  HipY  := LFHip
  KneeY := LFKnee
  FootY := LFFoot
  toeY  := LFToeSensor

  HipZ  := RRHip
  KneeZ := RRKnee
  FootZ := RRFoot
  toeZ  := RRToeSensor

  HipA  := LPawHip
  KneeA := LPawKnee
  FootA := LPawFoot
  toeA  := LPawToeSensor

  HipB  := RFHip
  KneeB := RFKnee
  FootB := RFFoot
  toeB  := RFToeSensor

  HipC  := LRHip
  KneeC := LRKnee
  FootC := LRFoot
  toeC  := LRToeSensor
```

```
    FootC := LRFoot
    toeC  := LRToeSensor


''Caculating if you need to add, subtract or do nothing with moving the hip from
''it's starting position
    If StartX == FinishX
       DirectionX := 0
       LegStateX  := 1
    ELSEIf StartX => FinishX
       DirectionX := -1
    ELSEIf StartX =< FinishX
       DirectionX := 1

    If StartY == FinishY
       DirectionY := 0
       LegStateY  := 1
    ElseIf StartY => FinishY
       DirectionY := -1
    ElseIf StartY =< FinishY
       DirectionY := 1

    If StartZ == FinishZ
       DirectionZ := 0
       LegStateZ  := 1
    ElseIf StartZ => FinishZ
       DirectionZ := -1
    ElseIf StartZ =< FinishZ
       DirectionZ := 1



    MoveHipsXYZ

PUB MoveHipsXYZ
    Debug.str(string(13, "MoveHipsXYZ! :) "))
    Debug.Bin(ina[0..5], 6)                         ''Displays the state of the 6 leg touch
                                                    ''sensors.  A "1" means that it's touching


    Debug.str(string(13, "LegStateX "))
    Debug.dec(LegStateX)
    Debug.str(string(13, "LegStateY "))
    Debug.dec(LegStateY)
    Debug.str(string(13, "LegStateZ "))
    Debug.dec(LegStateZ)
    Debug.str(string(13, "LegStateXYZ "))
    Debug.dec(LegStateXYZ)

    LegStateXYZ := LegStateX + LegStateY + LegStateZ
    If LegStateXYZ == 3
```

```
LegStateXYZ := LegStateX + LegStateY + LegStateZ
If LegStateXYZ == 3
  Debug.str(string(13, "Hips XY&Z are in place; now going to lower feetXY&Z! :) "))
  LowerFeetXYZ


MoveHipX

PUB MoveHipX
Debug.str(string(13, "MoveHipX"))

If HipXPlacement == FinishX
  LegStateX  := 1
  Debug.str(string(13, "Hip X is done moving; skipping X to MoveHipsY! :) "))
  MoveHipY

HipXPlacement := HipXPlacement + DirectionX*2
PSC.SETPOS(HipX, Ramp, HipXPlacement)
'waitcnt(clkfreq/10 + cnt)

''This section checks to see if the Toe sensor is touching something and if it is,
''pulls the hip back a bit and raises the leg
If ina[toeX] == 1
 Debug.str(string(13, "Leg/pin toeX (RPawToeSensor) touching!!! :) "))
 HipXPlacement := HipXPlacement + (DirectionX* -10)
 KneeXPW := KneeXPW + 30
 KneeXPW <#= KneeHightMax
 Debug.str(string(13, "KneeXPW "))
 Debug.dec(KneeXPW)
 waitcnt(clkfreq/10 + cnt)

 IF KneeXPW => KneeHightMax

     Debug.str(string(13, "KneeHightMax reached, going to hook leg "))
     'Brings stuck foot to starting place
     PSC.SETPOS(HipX, Ramp, StartX)
     PSC.SETPOS(KneeX, Ramp, 1100)
     PSC.SETPOS(FootX, Ramp, 800)
     waitcnt(clkfreq/1 + cnt)
    repeat
     If ina[toeX] == 1
        Debug.str(string(13, "Leg/pin toeX (RPawToeSensor) touching again "))
        Debug.str(string(13, "Going to Raise body "))
        'RaiseBody
        ''lifts body up so that foot or hip can go over or on top of an object
        Debug.str(string(13, "RaiseBody"))

        BodyHeight := BodyHeight - 30
        Debug.str(string(13, "BodyHeight"))
        Debug.dec(BodyHeight)
```

```
            Debug.str(string(13, "RaiseBody"))

            BodyHeight := BodyHeight - 30
            Debug.str(string(13, "BodyHeight"))
            Debug.dec(BodyHeight)

            If BodyHeight =< BodyHeightMax
              Debug.str(string(13, "BackUpAndChangeApproach")) 'Going to try and br
              BackUpAndChangeApproach 'HookLegOver
            ELSE
              Debug.str(string(13, "Raising Body"))
              PSC.SETPOS(KneeA, Ramp, BodyHeight)
              PSC.SETPOS(KneeB, Ramp, BodyHeight)
              PSC.SETPOS(KneeC, Ramp, BodyHeight)

              PSC.SETPOS(FootA, Ramp, BodyHeight)
              PSC.SETPOS(FootB, Ramp, BodyHeight)
              PSC.SETPOS(FootC, Ramp, BodyHeight)
             'MoveHipX

            HipXPlacement := HipXPlacement + DirectionX*2
            PSC.SETPOS(HipX, Ramp, HipXPlacement)

            If HipXPlacement == FinishX
              LegStateX  := 1
              Debug.str(string(13, "Hip X is done moving; skipping X to MoveHipsY!
              MoveHipY


      'Moves leg back and up
      PSC.SETPOS(HipX, Ramp, HipXPlacement)
      PSC.SETPOS(KneeX, Ramp, KneeXPW)
      PSC.SETPOS(FootX, Ramp, KneeXPW)
      'waitcnt(clkfreq/10 + cnt)

    MoveHipY


'   if ina[LFToeSensor] == 1
'     Debug.str(string(13, "Leg/pin LFToeSensor touching!!! :) "))

PUB MoveHipY
    Debug.str(string(13, "MoveHipY"))



    MoveHipZ

PUB MoveHipZ
    Debug.str(string(13, "MoveHipZ"))
```

```
PUB MoveHipZ
    Debug.str(string(13, "MoveHipZ"))
    'waitcnt(clkfreq/1 + cnt)



    MoveHipsXYZ
PUB  RaiseBody
''lifts body up so that foot or hip can go over or on top of an object
     Debug.str(string(13, "RaiseBody"))

     BodyHeight := BodyHeight - 30
     Debug.str(string(13, "BodyHeight"))
     Debug.dec(BodyHeight)

     If BodyHeight =< BodyHeightMax
         Debug.str(string(13, "BackUpAndChangeApproach")) 'Going to try and bring leg
         BackUpAndChangeApproach 'HookLegOver
     ELSE
         Debug.str(string(13, "Raising Body"))
         PSC.SETPOS(KneeA, Ramp, BodyHeight)
         PSC.SETPOS(KneeB, Ramp, BodyHeight)
         PSC.SETPOS(KneeC, Ramp, BodyHeight)

         PSC.SETPOS(FootA, Ramp, BodyHeight)
         PSC.SETPOS(FootB, Ramp, BodyHeight)
         PSC.SETPOS(FootC, Ramp, BodyHeight)

       MoveHipX

PUB  LowerBody
''Lowers body so that it's not so tall, goes back to normal height
     Debug.str(string(13, "LowerBody"))




PUB LowerFeetXYZ
''Lowers feet until they touch something
    Debug.str(string(13, "LowerFeetXYZ"))




PUB HookLegOver | StuckKneePW, StuckFootPW
'' Robot brings leg back to the start location and trys to hook it over the obstical
'' Last ditch attempt before going in a different direction
```

```
PUB HookLegOver | StuckKneePW, StuckFootPW
'' Robot brings leg back to the start location and trys to hook it over the obstical
'' Last ditch attempt before going in a differnt direction

 'Brings hip back to starting point with foot and knee at Max, then trys to go to go to Finish
 'with bringing foot up (lower PWs) to avoide obsticals while checking to see if foot is touching
 'something
    Debug.str(string(13, "HookLegOver"))

    StuckKneePW := 1100
    StuckFootPW := KneeHightMax

    'Brings stuck foot to starting place
    PSC.SETPOS(StuckLegHip, Ramp, StartX)
    PSC.SETPOS(StuckLegKnee, Ramp, StuckKneePW)
    PSC.SETPOS(StuckLegFoot, Ramp, StuckFootPW)
    waitcnt(clkfreq/1 + cnt)

     IF LegStateX

     StuckLegHip := StuckLegHip + StuckLegDirection

          'Records the "Stuck" legs variables in case the program needs to go to "Hook leg over"
           StuckLegHip          := HipX
           StuckLegKnee         := KneeX
           StuckLegFoot         := FootX
           StuckLegHipStart     := StartX
           StuckLegHipFinish    := FinishX
           StuckLegToeSensor    := ToeX
           StuckLegDirection    := DirectionX
           'StuckLegState        := LegStateX


'            StuckLegToeSensor
'           StuckLegHip          := HipX
'           StuckLegHipStart     := StartX
'           StuckLegHipFinish    := FinishX




PUB BackUpAndChangeApproach
'' Robot backs up a little and trys again
    Debug.str(string(13, "BackUpAndChangeApproach"))
```

1. Master direction angle is given from Ping))) sensor or other source

2nd Each leg unit takes the Master angle and add's its modifier based on the angle that the leg unit is at relative to the Master angle.

3rd. Based on the compound angle, the proportional amount of hip and knee movement and direction that is required to make that angle of movement.

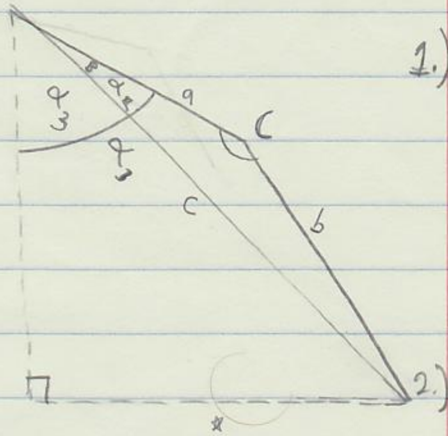
4th The amount of hip & knee movement is then modified based on the calculated distance the foot placement is from the hip's point of rotation.

The placement of the foot and the angles of the knee & hip vertical can be modified based on:

1) if the foot makes contact with something

2.) if the gyros require more or less extention in order to maintain a level body.

```
PUB PathCheck    '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.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
```



$c^2 = a^2 + b^2 - 2ab\cos C$

1.) $c = \sqrt{a^2 + b^2 - 2ab\cos C}$

$\frac{b}{\sin B} = \frac{c}{\sin C}$ / $\frac{\sin B}{b} = \frac{\sin C}{c}$ / $B = \arcsin\left(\frac{b \sin C}{c}\right)$

or use this one

2.) $B = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$

3.) $\alpha_2 = \alpha_3 - \alpha_1$

4.) $\frac{c}{\sin 90} = \frac{\star}{\sin \alpha_3}$ / $\star = \sin \alpha_3 \left(\frac{c}{\sin 90}\right)$

$\star = \sin \alpha_3 \cdot \frac{c}{1}$

this gives us the distance that the foot is away from the hip pivot and what the ajustment in the amount of hip movement (if any) has to be.

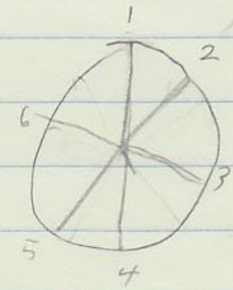
hip movement ajustment is equal to: $\left(\frac{\text{femur length}}{\star}\right) \times$ hip movement

s.
555286

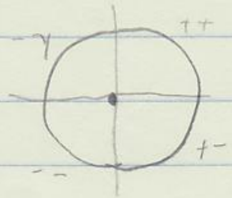

```
'Triginomitry funtion testing program
'
'
CON

  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000
VAR

  Long  sinn
  Long  angle
  Long  radius
  Long  x


OBJ
  'SL       : "SL32_INTEngine"
  F       : "Float32Full"
  Debug   : "FullDuplexSerialPlus"
  'fmath    : "FloatMath"
  fString : "FloatString"

PUB Start  | angleB1degrees, sideA, angleA, sideB, angleB2, angleB2degrees, sideC, angleC,
angleB3, angleB3degrees, legReach
  Debug.Start(31, 30, 0, 57600)
  F.Start

  waitcnt(clkfreq*4 + cnt)

  'Seed values for debuging and testing
   angleB1degrees := 110.0
   sideA  := 20.0
   sideB  := 35.588
   angleC := 40.0

   angleC := F.Radians(angleC)

  'Step one
  'For step "one" in caculating the distance the foot is from the point directly below the
   'hip's piviot point
   sideC := f.FSqr(f.FSub(f.FAdd(f.FMul(sideA, sideA), f.FMul(sideB, sideB)),
   f.FMul(2.0, f.FMul(sideA, f.FMul(sideB, f.Cos(angleC))))))
   Debug.tx(Debug#CR)                                  'Gives a charage return
   Debug.Str(String(13, "Side C "))
   Debug.str(fstring.FloatToString(sideC))

  'Step two
  'Use arccos instead of the law of sins to avoide the ambiguous case (que scary musice ;)
   angleB2 := f.ACos(f.FDiv((f.FSub(f.FAdd(f.FMul(sideA, sideA), f.FMul(sideC, sideC)),
   f.FMul(sideB, sideB))), (f.FMul(2.0, f.FMul(sideA, sideC)))))
   angleB2degrees := f.Degrees(angleB2)
```

```
 'Step two
 'Use arccos instead of the law of sins to avoide the ambiguous case (que scary musice ;)
  angleB2 := f.ACos(f.FDiv((f.FSub(f.FAdd(f.FMul(sideA, sideA), f.FMul(sideC, sideC)),
  f.FMul(sideB, sideB))), (f.FMul(2.0, f.FMul(sideA, sideC)))))
  angleB2degrees := f.Degrees(angleB2)
  Debug.tx(Debug#CR)                                       'Gives a charage return
  Debug.Str(String(13, "Angle B2 "))
  Debug.str(fstring.FloatToString(angleB2degrees))

 'Step three
 'Find angle "B3" using "B1" ("up and down" angle that we gave to the femur) and the
  '"B2" that we just found
  angleB3degrees := f.FSub(angleB1degrees, angleB2degrees)
  Debug.tx(Debug#CR)                                       'Gives a charage return
  Debug.Str(String(13, "Angle B3 "))
  Debug.str(fstring.FloatToString(angleB3degrees))

 'Step four
 'Find the reach of the leg (distance from the hip's piviot point to the point that the
 'leg touches the ground
 'by using a right riangle and the law of sin
  angleB3 := F.Radians(angleB3degrees)
  legReach := f.FMul(sideC, f.sin(angleB3))
  Debug.Str(String(13, "legReach "))
  Debug.str(fstring.FloatToString(legReach))




{
 'sideC := f.FSqr(f.FAdd(f.FMul(sideA, sideA), f.FMul(sideB, sideB)))
 'Debug.tx(Debug#CR)                                        'Gives a charage return
 'debug.str(fstring.FloatToString(sideC))


 'sideC := f.FMul(2.0, f.FMul(sideA, f.FMul(sideB, f.Cos(angleC))))
 'Debug.tx(Debug#CR)                                        'Gives a charage return
 'debug.str(fstring.FloatToString(sideC))


 sinn := SL.sin(30.0, 1)
 Debug.Str(String(13, "sin"))
 Debug.tx(Debug#CR)                                        'Gives a charage return

 debug.str(fstring.FloatToString(sinn))

 angleB := f.ASin(f.FDiv((f.FMul(sideB, f.Sin(angleC))), sideC))

 x    := F.Radians(40.0)
```

0 all hip
1



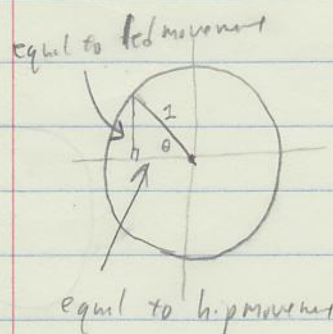
6)360 = 60



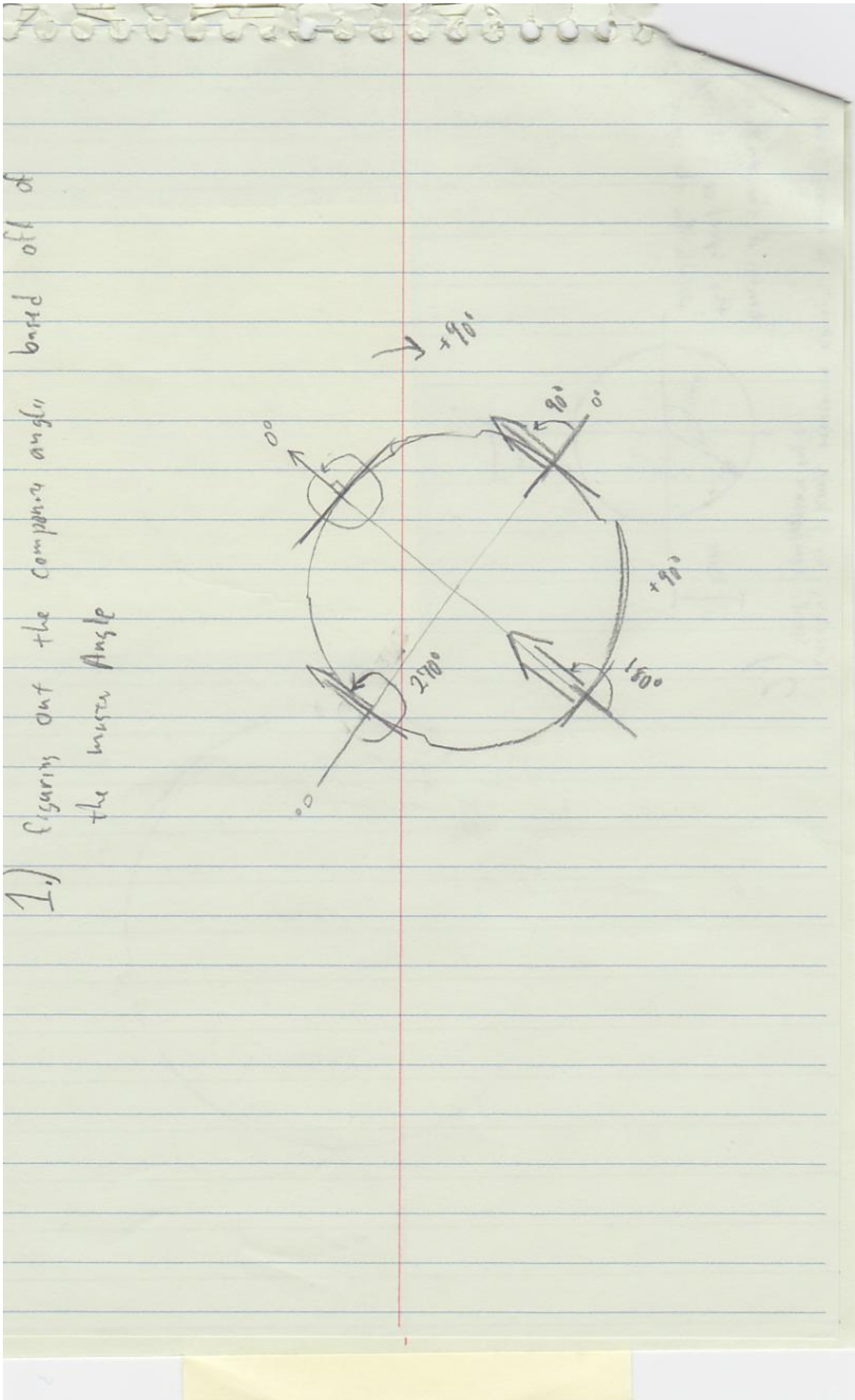
Direction is given in degrees, each leg unit then adds it's ajustment

thy complet to directn

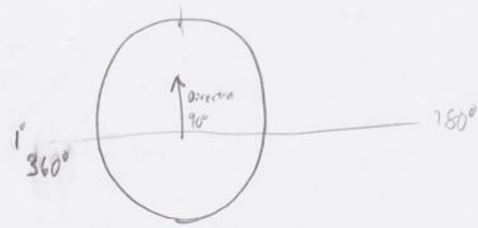
leg 1 = 90°
+60
leg 2 = 150°
+60
leg 3 = 210°
+60
leg 4 = 270°
+60
leg 5 = 330°
+60
leg 6 = 30°



Caculate the x, y cordinates (x is hip movement, y is tibia movement) the positive & negative signs lets you know if it's pulling or pushing


1.) figuring out the component angles based off of the master Angle
0°
0°
0°
90°
+90°
+90°
180°
270°

The direction determines the amount of movement is in the "foot" and if its "pulling" or pushing



Basic Determinance of hip & leg values

Direction value (DV) = 1 to 360.

If DV is greater than 180°, leg is pushing (start position == end position)
DV == DV-180 (This make the Value between 1 and 180)
If DV is greater than 90 Then DV == 179-DV (This give a value for DV between 1 and 90)
Amount of leg movement is equal to $\left(\text{speed}[1 \text{ to } 75] \cdot \frac{DV}{90} \right)$

[This amount of movement is then subtracted to the foot "start position" for the end value. It's also used determine how much is in the hip. (It's whatever is left over)]

Amount of hip movement is equal to Speed − leg movement
[This amount of movement is then subtracted from the hip "starting position"]

Now we need to figure out how much distortion (if any) based on the ending point of the knee. (If it's more or less than a 90° angle for the leg

Ajust foot & hip values for distortion prior to caculating starting and ending positions or the number of time the joint is moved 1°

[Use Combined side & bottom Sensor] by?

During leg retention phase

if the foot hit something on the side [Double check. leaned over?]

- lift leg higher & try again, repeat until success or leg height reaches its max

- if max is reached then try left & right of initial position @ original height

During leg lowering

lower until it touches something, double check that the foot is on firm ground by checking it again after x time [increase x in proportion to the closeness that the foot placement is @ the position that the ideal position was calculated]

Now use the positions that the legs find themselves @ to recalculate the amount of movement is required to go in the Master Direction angle

Speed is sacrificed to maintain Direction

Acknowledgments

I thank my wife for her love and support in everything