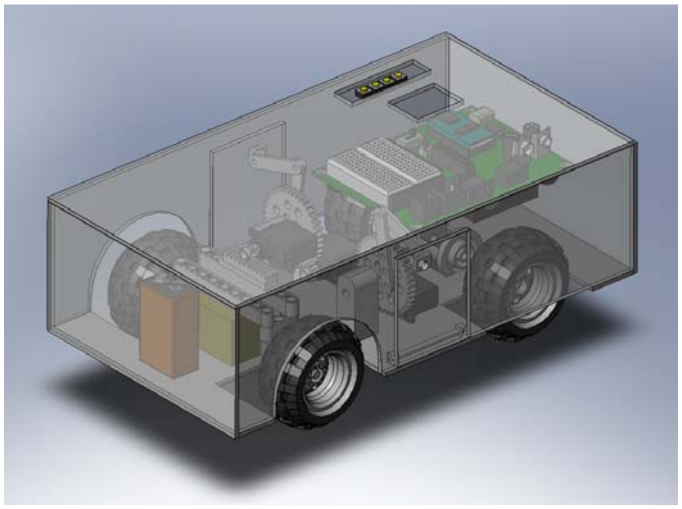
Mechatronics Final Project Report
Prof. Kapila

Flipper the Persistent Vehicle (Self-Uprighting Robot)

Christopher Clinton
Michael Litvinov
Ildi Telegrafi

5.19.08



Abstract

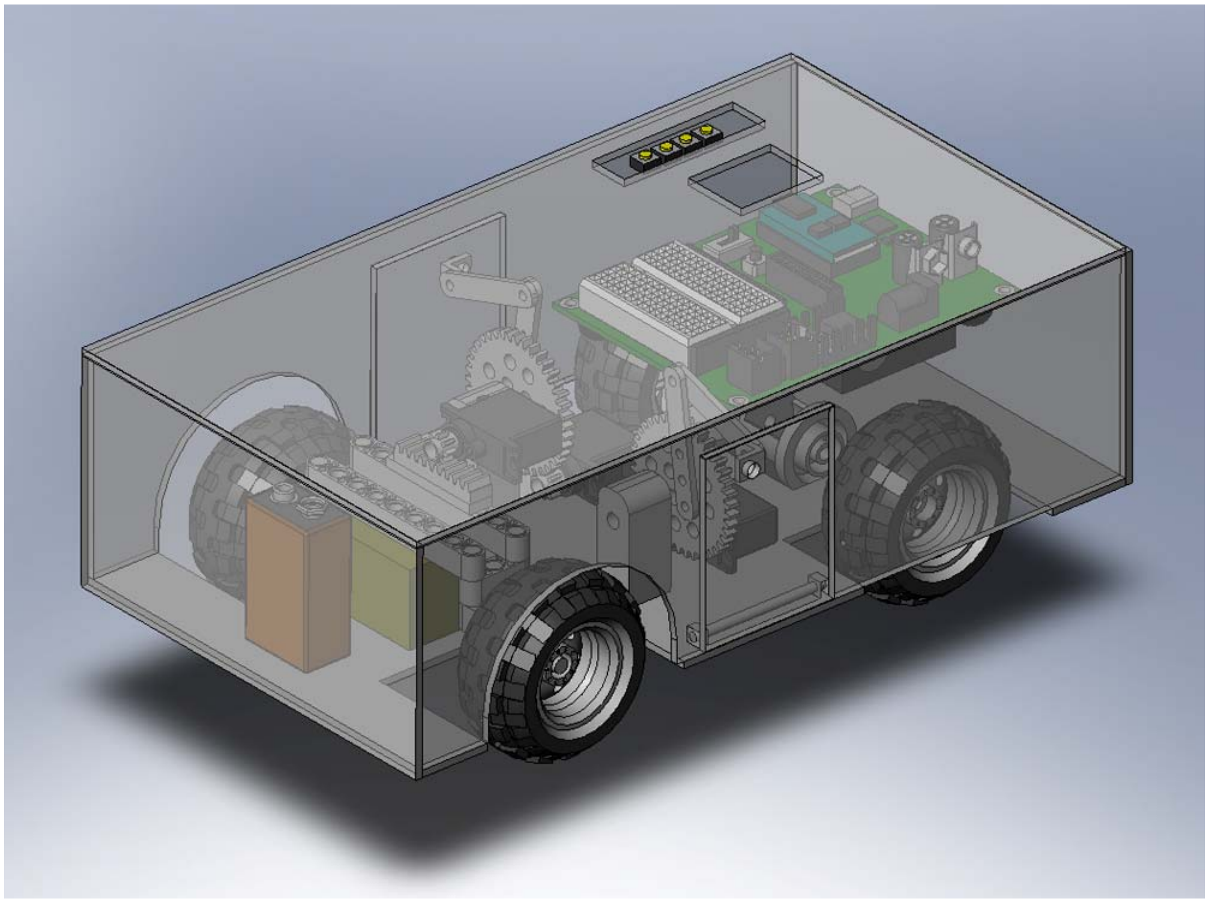
In this paper we present a practical approach to dealing with unexpected occurrences in vehicles and robots. Often times a vehicle or robot may turn over on its side and will no longer be able to perform its task without human aid. Through the use of the Basic Stamp 2 a robot was created that can autonomously detect when it is on its side and turn back over. This way the robot can continue with its main function (which in this case is to stay on its original course). This concept relies on the use of servos, accelerometer and compass.

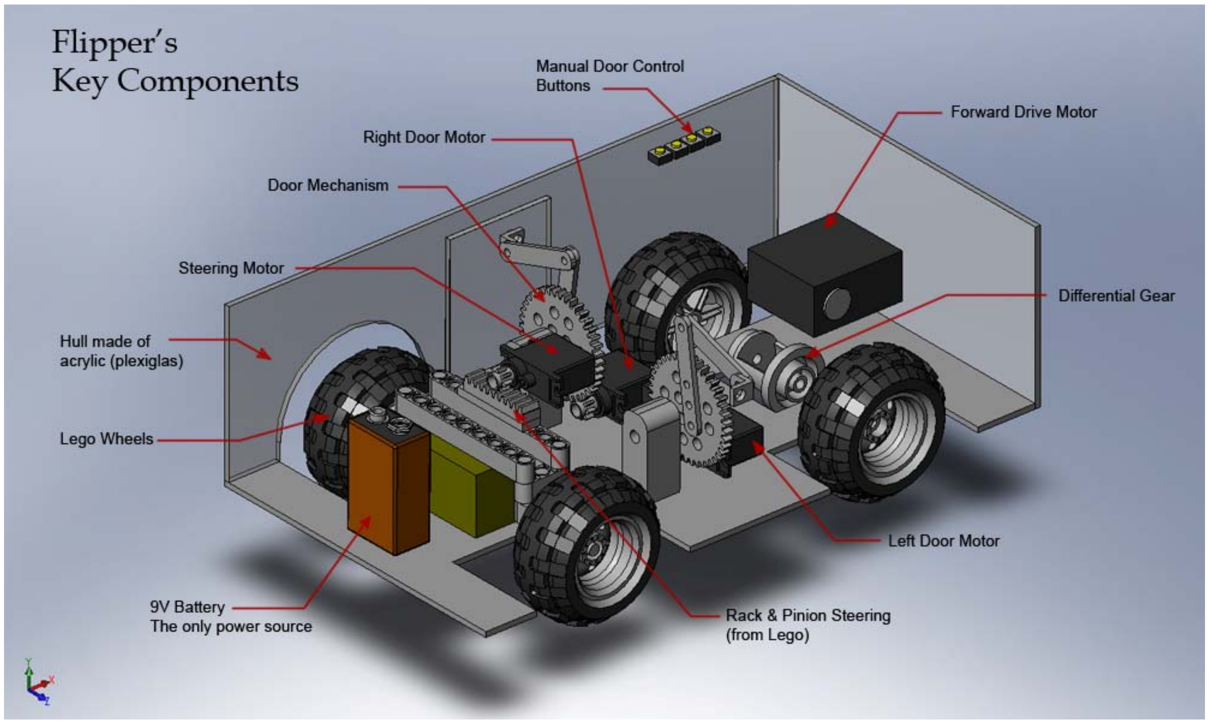
Introduction

Today, many robotic vehicles are used for various tasks. However, the design for these vehicles focuses mainly on what is needed to complete the objective. Unfortunately, this often disregards the randomness that comes with human interaction and the natural environment. NASA's Mars Rover was designed to traverse the rugged terrain on Mars. However, if something unexpected occurred and the vehicle turned over it would be rendered ineffective. To resume its mission, the vehicle must turn itself upright. This can be accomplished by activating a set of motors whose purpose is to rotate an arm about an axis parallel to the ground and thus push against it causing the robot to turn upright. This arm applies a force against the ground that causes the robot to move out of its current equilibrium point. Then it uses its own weight to level itself and enter the proper equilibrium position where both sets of wheels make contact with the ground. With all wheels touching the ground the vehicle can continue on its course. In this way the vehicle will be able to detect, autonomously, that its orientation is incorrect and remedy the situation.

Theory and Design

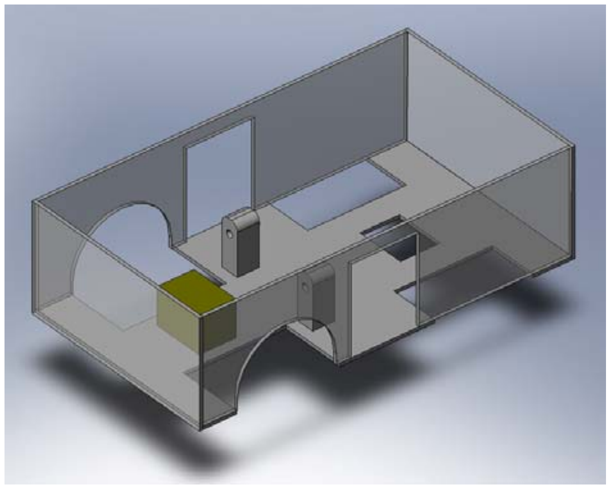
- **Schematic of key components**





- **Hull**

The hull of the Flipper (name that was given to our robot) was built out of plates of acrylic (plexiglas), which was cut using Dremel[1] tool and glued together with Gorilla Glue in places where strong adhesion is desired, and with Glue Gun[2] where non-permanent hold is needed.


Flipper's Hull

- **Sensors**

Flipper uses two digital sensors:

- Memsic Dual-Axis Accelerometer and
- Hitachi HM55B Compass.

The purpose of accelerometer is to sense whether or not the robot is tilted. A roughly 80° tilt is considered critical. Although Memsic can measure X and Y tilts, only X axis was needed for the purpose of this project.

Compass is used to give Flipper a purpose – imitation of a particular task that is given to a robot.[3] When the robot is turned on the value that's given by the compass is used as a reference heading which our robot will follow even if it's turned off course. Parallax sample code was used to operate the compass and only here – the rest of the program was hand typed. Standard heading values of 0 to 359 are used as the signal provided by the sensor.

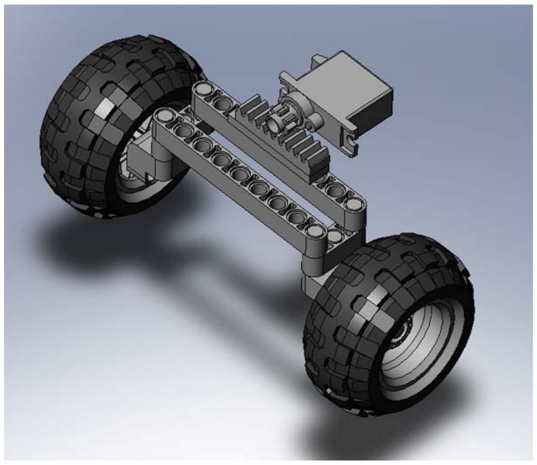
[1] Dremel is the brand name of a versatile rotary tool that can be used for light cutting and grinding jobs.
[2] Glue Gun uses a heating element to melt and extrude plastic, which cools down in a matter of seconds providing relatively strong adhesion.
[3] The idea of this project is to demonstrate that a robot can continue performing a task even if it's flipped over.

- **Actuators**

There are 4 servo motors used in the robot. 1 is the Continuous Rotation servo that came with our robotics kit and its purpose is to provide forward drive to the rear wheels via a differential gear. The other 3 motors are Parallax Pico servos which are smaller but powerful enough to accomplish the tasks we've given them.

Two of the servos were modified to provide continuous rotation to the gear sets that drive the doors[4]. The last, unmodified standard Pico servo is used for steering the robot. Rack and pinion transmission from a Lego set is used here.

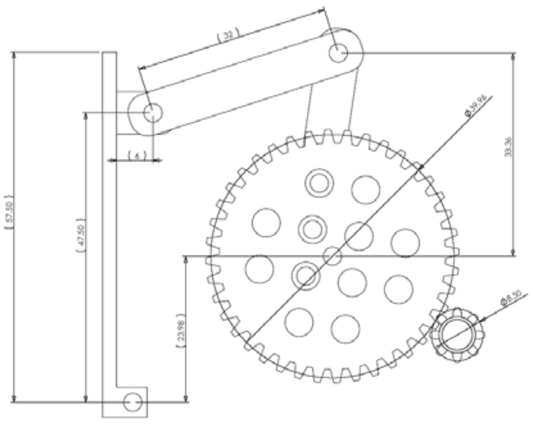
In order to interface the Parallax servo with a Lego gear, the gear's insides were drilled out to provide a firm fit onto the servo's axel.



Steering Mechanism

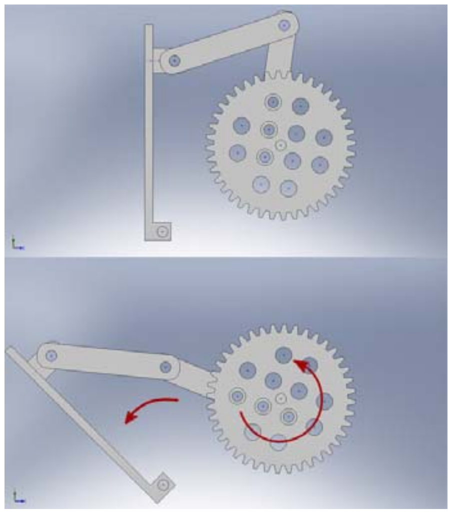
- **Featured Mechanism**

This is the main part of the project and has undergone a few changes from its original concept. The idea was to make a custom part that would slide down and then turn to flip the robot. This however turned out to be an impossibility as rapid prototyping at poly is not what it's hyped to be. Custom part was not an option and a simpler work-around was thought up. Latter involves a large gear that is attached to a lever system which, when the gear is rotated, pushes the door out with enough force to tip over the robot when it's on a side. The levers were machined out of aluminum and bolted



[4] Doors is the term that will be used throughout the report to denote the side panels that perform the task of uprighting the robot.

together to provide a flexible joint. The large gear is driven by a smaller gear attached to a Pico servo. The gear ratio creates a lot of torque which is translated into quasi-linear motion.



The doors are hinged to sides of the hull and are size-fitted to provide smooth operation

Opening and closing range of doors with and without load (where load is the weight of the robot) vary drastically – this is attributed to the fact that the servos have been modified and have no way of knowing their current position (unmodified servos could have been used, but they needed to have continuous rotation for the original design, which did not work out). This was remedied by tweaking the code to "flip out" more than "flip in". Also, 4 buttons were added to manually adjust the door position if needed.

Another obstacle associated with normal operation of the door mechanisms is the fact that all servos "twitch" when the robot is turned on. This is due to the fact that the servos are directly connected to Vdd, and understand it as a first (long) pulse. The servo rotates just a little and then waits for normal pulses associated with Pulse Width Modulation required to drive the servos. However small the twitch is, it was a nuisance when testing and debugging the robot. A fix was to make the servos rotate just a bit in the direction opposite to the "twitch" upon initialization. Another possible (perhaps better) fix would be to use a transistor to turn on the motors when needed. With limited space onboard the Flipper this option was set aside but is noteworthy.

- **Speaker**

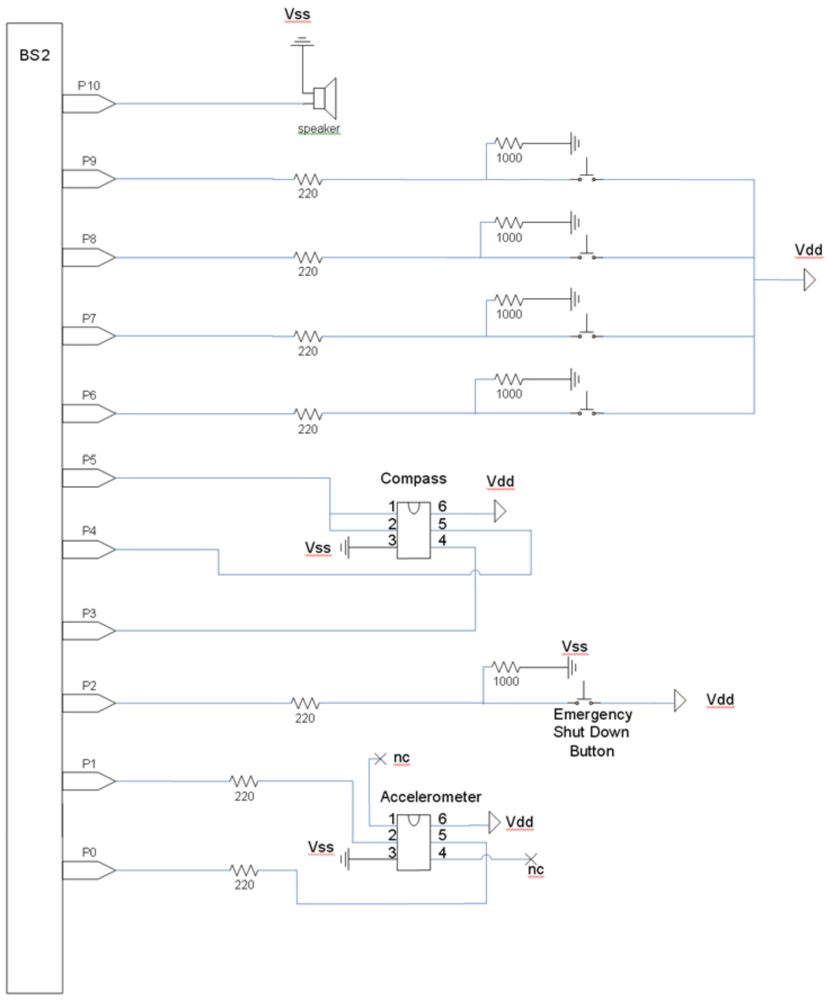
Flipper features a piezo speaker that is entirely unnecessary but was a fun addition with huge potential. Currently it plays two "melodies": one upon detecting critical tilt, and another upon regaining the original heading. It is driven using PULSOUT command with varying pulse durations for frequency change. Actual melodies could be programmed providing there was sufficient enough time and lack of better things to do.

- **Buttons**

There are 5 external buttons on the robot available for the user to press. One button is the emergency shut down – once it is pressed, the program will END and the robot will stop. It can come in handy if there is a gear stuck, or quick turn off is required for any other reason.

The other 4 buttons control the doors. Two are for the right door and two are for the left door. One button on each pair opens the door and the other closes it. This is useful for demonstration, making sure the doors are in working order, and for manual adjustment if the door doesn't close completely.

- **Circuit**



- **Analysis**

The main advantage of this device over others is its ability to continue its mission even when environmental effects cause it to flip over. When other devices are turned over on their sides they are rendered inoperable and can no longer complete their task. This device can overcome some of those obstacles and continue on. Applications range from armored trucks to extraterrestrial rovers. It is truly surprising that the idea has not been implemented.

A disadvantage of this device is its inability to physically turn itself upright when it is on its top surface. This is due to the fact that it has no overturning mechanism on its roof. Adding one would require more space, parts, and labor and was deemed omissible since two sides are enough to demonstrate the concept.

The cost of the device is approximately $400. This includes 4 motors, plexiglas, electronic components, accelerometer, compass, battery connector, bolts, glue, and miscellaneous tools.

Mass production cost may vary based on field of application and complexity of the design. To add a flipping capability to one side of a robot or a vehicle, the manufacturer would need an electromotor, a hinged plate (in armored truck's case), a gear box or possibly an oil pressure cylinder to drive the door. In some cases the mechanism could be activated by the driver of the vehicle and no orientation detection would be needed in that case.

- **Code**

```
' {$STAMP BS2}
' {$PBASIC 2.5}

' ***************  FLIPPER THE PERSISTENT VEHICLE (aka OVERTURNER) *******************

' **** Compass Pins/Constants/Variables from Parallax ***
DinDout      PIN     5                                  ' P5 transceives to/from Din/Dout
Clk          PIN     3                                  ' P3 sends pulses to HM55B's Clk
En           PIN     4                                  ' P4 controls HM55B's /EN(ABLE)

Reset        CON     %0000                              ' Reset command for HM55B
Measure      CON     %1000                              ' Start measurement command
Report       CON     %1100                              ' Get status/axis values command
Ready        CON     %1100                              ' 11 -> Done, 00 -> no errors
NegMask      CON     %1111100000000000                  ' For 11-bit negative to 16-bits

xcom         VAR     Word                               ' x-axis data
ycom         VAR     Word                               ' y-axis data
status       VAR     Nib                                ' Status flags
angle        VAR     Word                               ' Store angle measurement

'**** Accelerometer Pins/vars

xTilt        VAR     Word                               ' X tilt value
'yTilt       VAR     Word                               ' Y tilt value NOT USED
xTpin        PIN     0                                  ' X pin of accelerometer
yTpin        PIN     1                                  ' Y pin of accelerometer

'**** Motor Pins
Steer        PIN     12
Drive        PIN     13
Right        PIN     14
Left         PIN     15


'**** Miscellaneous Variables
i            VAR     Word                                     ‘ temp variable
a            VAR     Word                                     ' a=angle-TargetHeading
b            VAR     Word                                     ' b=angle-TargetHeading+360
TargetHeading  VAR     Word                                   ' Target heading
Window       VAR     Word                                    ' allowable deviation from course
Window = 20
Played       VAR     Bit                                      ' for sounds
Played =1
```

```
GOSUB GoStraight                                          ' Straighten Steering


' *** Get Current Heading
GOSUB CheckDirection                                       ' Call compass sensor
  TargetHeading = angle                                    ' Set this heading as Target

DEBUG DEC TargetHeading


'** auto adjust flippers to negate the "servo start up twitch"

' Flip In  right
  FOR i=1 TO 1
    PULSOUT Right, 900
    PAUSE 20
  NEXT

' Flip Out left
  FOR i=1 TO 1
    PULSOUT Left, 1000
    PAUSE 20
  NEXT

'******* MAIN OPERATION LOOP ****************************************

DO

 IF IN2 = 1 THEN                                          ' emergency shut down button
  END
 ENDIF






'***** door adjustment buttons
IF IN6 = 1 THEN
                       ' RIGHT Flip Out
  FOR i=1 TO 1
    PULSOUT Right, 100
    PAUSE 18
  NEXT
```

```
ELSEIF IN7 = 1 THEN
                                   ' RIGHT Flip In
  FOR i=1 TO 1
    PULSOUT Right, 1500
    PAUSE 20
  NEXT

ELSEIF IN8 = 1 THEN
                                   ' LEFT Flip Out
  FOR i=1 TO 1
    PULSOUT Left, 100
    PAUSE 18
  NEXT


ELSEIF IN9 = 1 THEN
                                   ' LEFT Flip In
  FOR i=1 TO 1
    PULSOUT Left, 1500
    PAUSE 20
  NEXT

  ENDIF

  ' *** end buttons

PULSIN xTpin,1,xTilt                    ' get x tilt

IF xTilt > 3000 THEN                    ' Robot is on its Right side
  GOSUB FlipRight
ELSEIF xTilt < 2000 THEN                    ' Robot is on its Left side
  GOSUB FlipLeft
ENDIF




' ********* TURNING ****************
GOSUB CheckDirection                              ' Where am I going?

a= angle-TargetHeading                            ' temporary vars for evaluating absolute values (see trueAbs)
b= angle-TargetHeading+360

GOSUB trueAbs                                     ' evaluates absolute value (with negatives)
```

```
IF a < Window THEN                                      ' Heading is within allowable window of deviation

  IF Played = 0 THEN
    GOSUB OnCourseSound
    Played = 1
  ENDIF

ELSEIF a < b  THEN                                      ' crazy code
    Played = 0
   IF(angle<TargetHeading) THEN
    GOSUB TurnRight
   ELSE
    GOSUB TurnLeft
   ENDIF
 ELSEIF(angle<TargetHeading) THEN
    GOSUB TurnLeft
ELSE
   Played = 0
  GOSUB TurnRight

ENDIF

' ********* end of turning ***************

GOSUB MoveForward                                        ' move on!

LOOP


' *** main operation loop ends here ***************

' ******************************************************************
' ************ NEXT PAGE FOR SUBROUTINES *****************************************************






' *************** Subroutines ************************************
trueAbs:
 IF a>32767 THEN
 a=65535-a
 ENDIF
 RETURN

GoStraight:                    ' position front wheels facing forward
```

```
  FOR i=1 TO 50
    PULSOUT Steer, 750
    PAUSE 10
  NEXT

RETURN
' ***************************

TurnRight:                          ' make a right turn

  FOR i = 1 TO 20                 ' steer right
    PULSOUT Steer, 1200
    PAUSE 20
  NEXT

  FOR i = 1 TO 30                 ' drive a bit forward
    PULSOUT Drive, 500
    PAUSE 20
  NEXT

  GOSUB GoStraight

RETURN
' ***************************

TurnLeft:                       ' make a left turn

  FOR i = 1 TO 20                 ' steer left
    PULSOUT Steer, 400
    PAUSE 20
  NEXT

  FOR i = 1 TO 30                 ' drive a bit forward
    PULSOUT Drive, 500
    PAUSE 20
  NEXT
  GOSUB GoStraight
RETURN
' ***************************
MoveForward:
  FOR i = 1 TO 10
    PULSOUT Drive, 500
    PAUSE 5
  NEXT
RETURN
' ***************************
```

```
FlipRight:

' Play sound
  GOSUB FlipSound

' Flip Out
  FOR i=1 TO 35
    PULSOUT Right, 100
    PAUSE 18
  NEXT

  PAUSE 1000

' Flip In
  FOR i=1 TO 20
    PULSOUT Right, 1500
    PAUSE 25
  NEXT

RETURN
' ***************************

FlipLeft:

' Play sound
  GOSUB FlipSound
                                    ' Flip Out
  FOR i=1 TO 25
    PULSOUT Left, 1500
    PAUSE 20
  NEXT
  PAUSE 1000
                                    ' Flip In
  FOR i=1 TO 26
    PULSOUT Left, 100
    PAUSE 20
  NEXT
RETURN
' ***************************
' **************** SOUNDS **************

' Sound weee wooo
FlipSound:
  FOR i=500 TO 100
    PULSOUT 10, i
  NEXT
  PAUSE 100
  FOR i=100 TO 500
    PULSOUT 10, i
  NEXT
RETURN

' pe lee beep beep
OnCourseSound:
FOR i=1 TO 70
  PULSOUT 10, 800
  NEXT

FOR i=1 TO 200
  PULSOUT 10, 200
  NEXT

FOR i=1 TO 40
  PULSOUT 10, 50
  NEXT
  PAUSE 100
  FOR i=1 TO 40
  PULSOUT 10, 50
  NEXT
RETURN

'*** Compass (code by Parallax)

CheckDirection:                             ' Compass module subroutine

  HIGH En: LOW En                           ' Send reset command to HM55B
  SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]

  HIGH En: LOW En                           ' HM55B start measurement command
  SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
  status = 0                                ' Clear previous status flags

  DO                                        ' Status flag checking loop
    HIGH En: LOW En                         ' Measurement status command
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN  DinDout,clk,MSBPOST,[Status\4]  ' Get Status
  LOOP UNTIL status = Ready                 ' Exit loop when status is ready

  SHIFTIN  DinDout,clk,MSBPOST,[xcom\11,ycom\11]   ' Get x & y axis values
  HIGH En                                   ' Disable module

  IF (ycom.BIT10 = 1) THEN ycom = ycom | NegMask  ' Store 11-bits as signed word
  IF (xcom.BIT10 = 1) THEN xcom = xcom | NegMask  ' Repeat for other axis
```

```
angle = xcom ATN -ycom              ' Convert x and y to brads
angle = angle */ 360                ' Convert brads to degrees

RETURN   ' RETURNS angle
```

- **Bill of Materials**

Basic Stamp & sensors	$300
Acrylic plates	$30
Custom machined parts	$30
Lego gears and wheels	$20
Extra servos	$20
Glue	$5
Bolts	$5
Battery Connector	$1
Joy of putting it all together	Priceless