Rollie: A Two-Wheeled Robot

William D. Sherman

botronics@earthlink.net

Abstract

The idea of building a robot with two wheels was tested. Requirements for this project included the use of readily available and inexpensive materials. The use of standard size hobby servos to move the robot, required weight to be minimized and distributed in order for the robot to maintain balance. The design included the use of sensors to detect obstacles and people. The possible development and potential problems in detecting a remote beacon or IR command transmission was fully explored.

1 Why Two Wheels?

Why build a robot (fig. 1) with just two wheels? The use of two wheels enables the robot to execute turns centered upon it axis. Such turning allows accurate scanning and maneuvering without using a large ground area. Tight turning is not possible with a fixed four-wheel design. The use of two wheels with an idler wheel located in the front or back of the robot to maintain balance was considered but found undesirable. The use of two wheels allows the robot's electronics to be protected inside the space between the wheels. Additionally, the height of the wheels gives the robot good ground clearance over obstacles. If the robot comes up against a wall it simply flips over, with the robots interior completely protected. Interrupting rotation while rolling causes a gentle rocking action of its interior, allowing scanning of the area ahead.

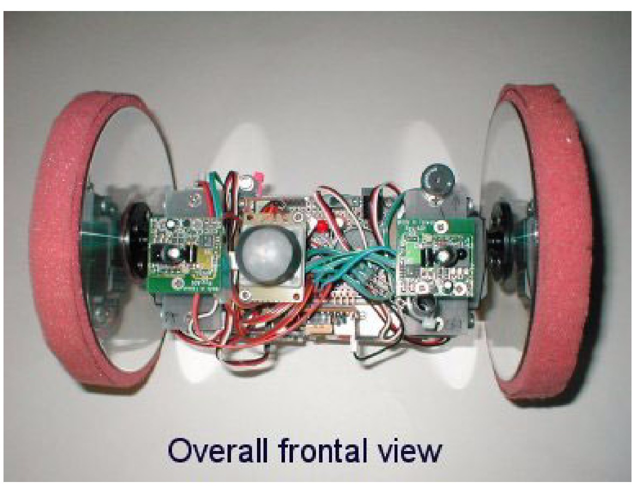


Fig. 1

2 Wheel Design

A simple wheel design, easily duplicated by others, was desired. Use of a CD-ROM disc was just the right size, occasionally found in one's mailbox as part of the junk mail everyone receives. The use of a single disc lacked the dimensional stability and strength required for a wheel. Using two discs with a ½ inch foam disc sandwiched between solved this problem. The foam disc (fig.2) is made ½ inch larger in diameter than the CDs, giving a ¼ inch edge for traction with the floor.

To hold this disc-foam-disc sandwich together, four ½ inch spacers were used to separate the CDs (fig. 6) while one side was attached to a round servo horn. Existing holes in the servo horn were opened in size to allow the use of 4-40 machine screws to hold everything in place. For increased traction, the edge could be scalloped, though this was not tried.

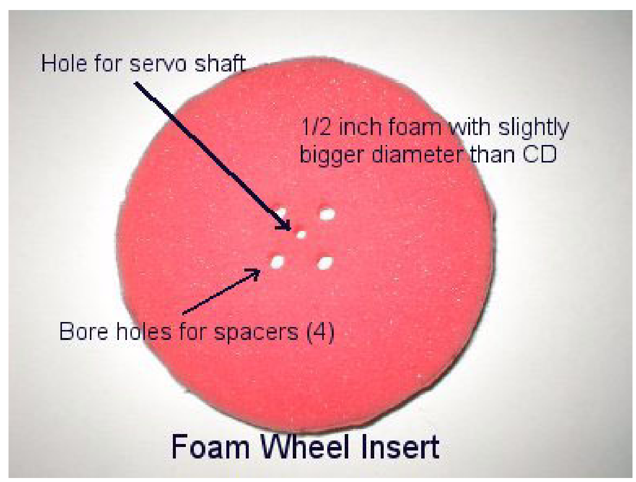


Fig. 2

3 Circuitry

3.1 Controller

A 16F84 PIC microcontroller from Microchip was selected for the robot's controller. The 16F84 is inexpensive, easily obtained, uses very little power and can be programmed "in-circuit" with the serial port of a computer. In-circuit programming allows programming of the PIC microcontroller without removing the device from the robot. Only four lines brought out to the PC serial port along with a 12v power source are all that is required to program the PIC. Programs were written and compiled in Basic using Micro Engineering Lab's PIC Basic Pro©, then downloaded using IC-Prog©, a freeware program found on the Internet.

The 16F84 flash memory is limited to 1024 words, while other PIC microcontrollers with more memory, faster speed and increased I/O lines can be used, the 16F84 running at 20 Mhz was adequate for this design. A small 1-7/8" by 2-3/4'' perf board from Radio Shack (274-150) contains the electronic circuit of the controller and is mounted in the center of the robot's base. Connection of the sensors to the circuit was accomplished by mounting female headers to the board. This allowed easy plug-in connections. The headers provide power, common, and input/output to the PIC's port. Mounted on each end of the base are the servos that provide rotation of the wheels.

3.2 Power

Four "AA" rechargeable alkaline batteries are used to provide power and are mounted under the base. The weight of the batteries keeps the center of gravity below the wheels' horizontal axis and acts as a pendulum in maintaining the position of the robot as it rolls along. The batteries supply power for about 90 minutes of autonomy to the robot.

4 Wheel Motors

4.1 Modified Servos

To power the wheels, standard hobby servos were used. These servos were modified for continuous rotation by taking apart and removing the mechanical connection of the position feedback potentiometer. This is a common practice with robot builders and several methods can be found on the Internet. Sending pulse width modulation to the servo from the robot's microcontroller controls rotational speed. Forward, backward and full stop are possible.

4.2 Power Saving Circuit

To prevent the servo's idle current of 8 milliamps each from putting a drain on batteries, a power saver circuit (fig. 3) was developed. The servos can be completely powered down when not needed. A IRF520 mosfet, used as a switch, was inserted on the "high side" of the positive power connection to the servo. A PVI5100 photovoltaic isolator (PVI) IC from International Rectifier was used to drive the mosfet. The PVI generates the turn-on voltage for the mosfet and eliminates the voltage drop that would occur when driving the mosfet in this fashion. Placement of the mosfet on the high side keeps the input voltage relationship with the microcontroller the same when off, further reducing idle current. A 10 megohm resistor between gate and source, quickly turns off the mosfet when the PVI is off.

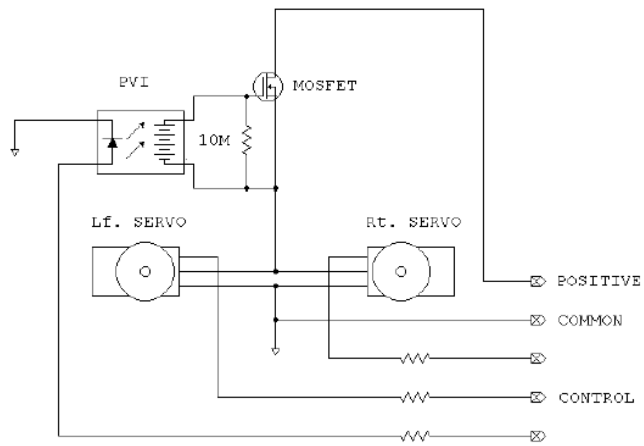


Fig. 3 Power Saver Circuit

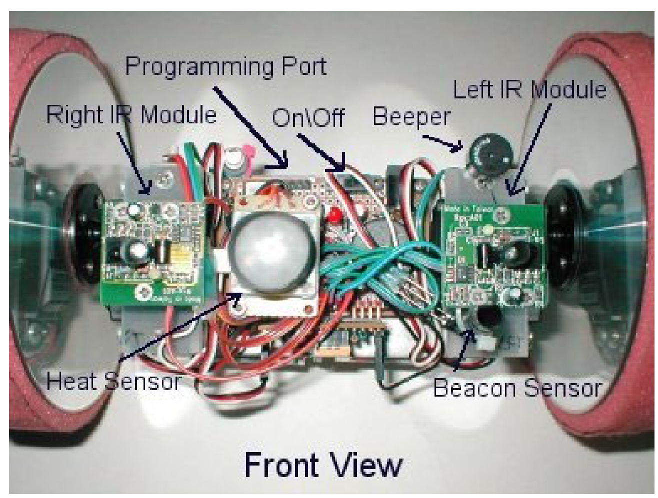


Fig. 4 Sensor Location

5 Sensors

5.1 Avoiding Obstacles

Obstacle avoidance is one of the primary activities of the robot when moving. Avoidance is achieved by emitting beams of IR light and detecting the reflected IR from objects with a sensor (fig. 4). To prevent interference from other sources of light, only light modulated at 38 kilohertz is emitted and received. An IR LED on each side of the robot is pulsed to high power with currents as high as 40ma. Pulses are activated by software as needed during maneuvering. Reception of reflected IR is accomplished by the use of a Panasonic PNA4602, made to respond only to 38 kilohertz. The range of detection depends on the reflectance of the obstacle and is in the range of 3 to 12 inches. The sensor, after detecting IR, outputs a low state and remains in this state briefly. This delay allows the microcontroller to pulse the IR LED, then check for the response of the sensor without doing both at the same time.

5.2 Passive Infrared

A low power, passive infrared (PIR) sensor is used to detect people or hot objects in the path of the robot. The PIR can only see objects that move. Scanning the area by rotating horizontally allows the robot to see people or other hot objects that are stationary. The relatively wide field of view is restricted by a section of black heat shrink tubing placed around the PIR. Restricting the field of view allows greater accuracy in locating targets of interest.

6 Use of a Beacon

6.1 Charging plate

The use of a beacon enabling the robot to seek-and-find was an idea to explore. Such a beacon could allow the robot to find its way “home” so as not to wander too far away. If the wheels of the robot were constructed of a conductive foam material and if the robot could make its way on top of a sectional metal plate, then the robot could draw power and charge it’s own batteries. The placement of this plate would need to be marked with a beacon for the robot to locate.

6.2 Communication

Information could be sent from the beacon to the robot for further instructions. Sending ASCII characters at 1200 baud was successful between two PIC microcontollers with programs written in Basic showing that this is possible. A range of about 15 feet between beacon and receiver was possible during testing of this idea. The implementation of communication with the robot itself has not been done at this time.

6.3 Multi-frequency Required

One problem that presented itself during testing was infrared from the beacon at the same frequency (38kc) interfered with the obstacle avoidance sensors of the robot. Using sensors operating at a different frequency of modulation should help with this problem. Panasonic IR sensors are available at other frequencies of 36.7, 40 and 56.9 kilohertz.

7 Remote Control

Some kind of remote control was desirable when the robot was off on its own during behavior tests. Sometimes the robot would wander into a dangerous situation and had to be “rescued”. Between IR obstacle avoidance routines the IR sensors are free to sense from other sources such as a remote control. If the robot receives IR during this time, the robot was programmed to turn right. With a little skill, an operator can steer the robot out of trouble. An IR remote (fig. 5) using the same pulse modulation frequency of 38 kilohertz was constructed with another 16F84 PIC and programmed in Basic.

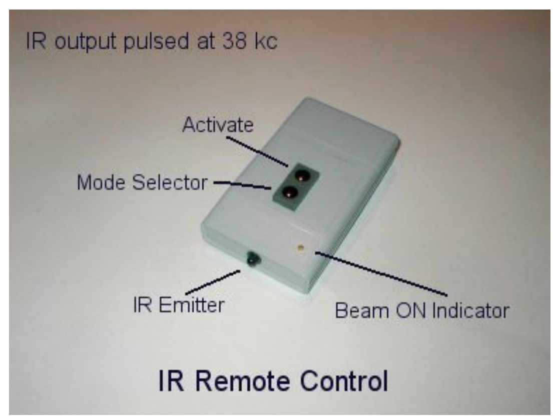


Fig. 5

8 Bells and Whistles

To help follow how the program progressed as it ran and for trouble shooting purposes, various auditory and visual indicators were added. Small green leds were added to each IR obstacle avoidance modules to indicate an object was encountered. A bright blue LED was useful to warn people of its prescience when the robot was allowed to interact with the pubic. A piezo beeper and speaker were added to beep at different points of the program. Children and adults alike found the beeping and flashing lights very attractive.

9 Behavior

9.1 Getting Around

The priority of the robot was to move forward, avoid obstacles, and move toward people. Before making a move the program sends a burst of 38 kilohertz IR from the IR LED's, then immediately checks the state of the IR receivers. If the way is clear, a forward motion is made. Forward motion continues until an object is encountered. If only the right or left IR sensor found something, then the robot would first backup a bit, followed by a right or left turn, depending on which sensor was activated. Backing up helps clear an area for a turn.

The program keeps track of the number of backups attempted before returning to forward motion. After a preset number of backups occur, the robot makes a right turn, in an attempt to get around an obstacle. Forward motion is counted as how many times the forward motion subroutine is accessed and saved as a variable. A reset of this count occurs on each backup, or turn.

9.2 Scanning

Once a clear path has been taken, the robot stops to scan for both the beacon and heat radiated by people. First rotation is made in quick steps clockwise and scans are made for the beacon. If the beacon is found, forward motion starts. After scanning for the beacon the robot then scans for heat, rotating counterclockwise back to its starting position. The robot pauses for a few seconds during each rotation to allow the PIR to react. If heat is detected forward motion starts. The forward motion counter variable resets to zero and forward motion continues until an object is encountered.

9.3 Watchdog

If no encounters have been made with objects and many scans have been completed, the robot takes a rest in "watchdog mode". During this time, all servos are turned off to conserve power, however the PIR sensor remains active. Upon detection of heat, the robot "wakes up", makes sounds, flashes the blue LED and starts to move forward. Holding down the mode switch while switching on the power, starts watchdog mode immediately, making the robot a handy security device.

10 Conclusion

Building a two-wheel robot provided a different and challenging way to maneuver a robot around. Maintaining balance was found to depend on weight distribution and proved to be no problem with this design. Changes in wheel diameter, weight distribution, and width of the base need to be addressed. Larger designs may require gearhead motors for greater torque and perhaps a method for sensing attitude for controlling balance. Students at the high school level or at a local robotics club could build the two-wheel robot design. Since the robot is made from materials that are commonly available, it is possible to build and program with very little investment.

Supplemental Photos

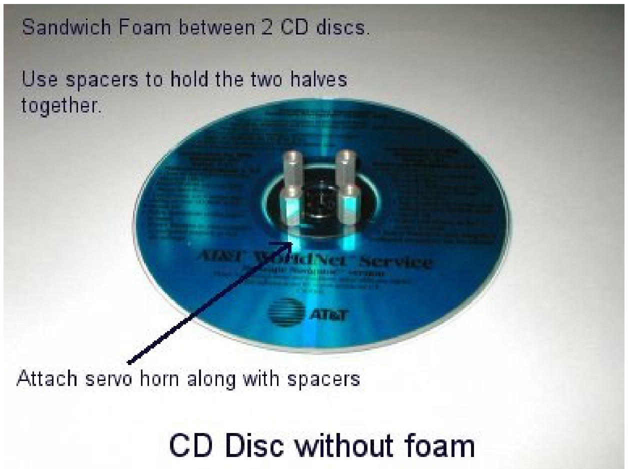


Fig. 6 Wheel Spacers mounted on CD disc

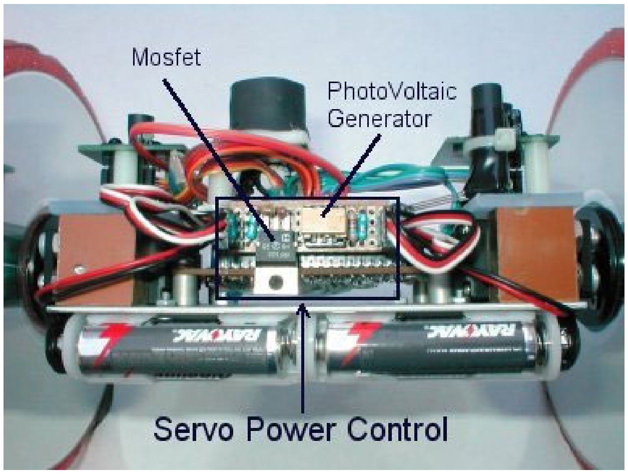
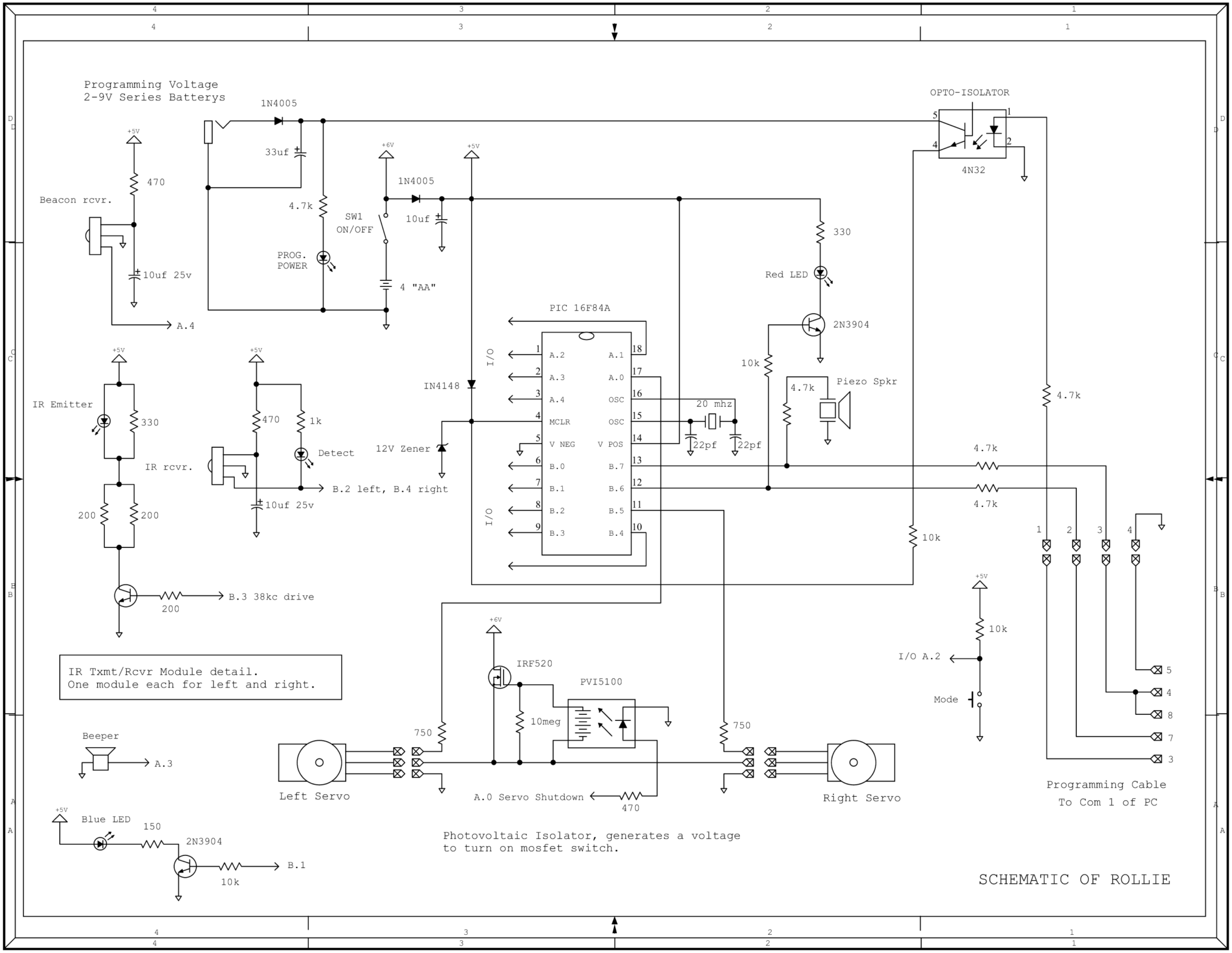


Fig. 7 Bottom View of Rollie


Programming Voltage
2-9V Series Batterys
1N4005
33uf
4.7k
PROG.
POWER
SW1
ON/OFF
10uf
4 "AA"
Beacon rcvr.
470
10uf 25v
A.4
OPTO-ISOLATOR
4N32
330
Red LED
2N3904
10k
Piezo Spkr
4.7k
PIC 16F84A
I/O
A.2
A.3
A.4
MCLR
V NEG
B.0
B.1
B.2
B.3
A.1
A.0
OSC
OSC
V POS
B.7
B.6
B.5
B.4
20 mhz
22pf
22pf
IN4148
12V Zener
IR Emitter
330
IR rcvr.
200
200
470
1k
Detect
B.2 left, B.4 right
10uf 25v
B.3 38kc drive
IR Txmt/Rcvr Module detail.
One module each for left and right.
Beeper
A.3
Blue LED
150
2N3904
10k
B.1
IRF520
PVI5100
10meg
750
750
Left Servo
Right Servo
A.0 Servo Shutdown
470
Photovoltaic Isolator, generates a voltage
to turn on mosfet switch.
10k
I/O A.2
Mode
Programming Cable
To Com 1 of PC
SCHEMATIC OF ROLLIE

```
'***************************************************************
'*  Program:  rollybeacon1.BAS                                 *
'*  Author:   Bill Sherman                                     *
'*  Notice:   Copyright (c) 2001 Bill Sherman                  *
'*            All Rights Reserved                              *
'*  Date:     6-17-02                                          *
'*  Version:  1.0                                              *
'*  Notes:    wheel speed con added                            *
'*            1021 words used                                  *
'***************************************************************
Include "modedefs.bas"
Define osc 20

TRISA=%11110100                 'set porta i/o as needed
TRISB=%00010101                 'set portb i/o as needed

beakin      var   byte          'charactor from beacon
rest        var   byte          'decide to rest var
y           var   byte          'scan count
f           var   byte          'forward movement count
s           var   byte          'backing up count
n           var   word          'for/next loop counter
righteye    var   portb.4       'right sensor
lefteye     var   portb.2       'left sensor
speed       con    20           'speed variable
pyro        var   portb.0       'pyro heat sensor
irrec       var   porta.4       'reciever for beacon, make different freq.
push        var   porta.2       'pushbutton switch
lwheel      con    764          'left servo wheel zero speed
rwheel      con    776          'right servo wheel zero speed

init:                           'initalize key system
    low portb.3                 'turn off ir
    low porta.3                 'beeper off
    s=0                         'backup count zero
    y=0                         'scan count zero
    f=0                         'set backing up count to 1
    rest = 0                    'clear rest var to zero
    gosub zero                  'stop servos

flash:                          'flash blue led, detect mode select, servos ON
    high porta.1                'turn on motor pwr
    high portb.1                'turn on blue led
    if push = 0 then watchdog   'Hold down ON button for watchdog mode
    pause 2000                  'keep blue led on for 2 sec
    low portb.1                 'turn off blue led
```

```
pulseme:                                'generate 38kc light
   high portb.1                         'blue led ON
   for n= 1 to 250                      'start burst 38 kc
   high portb.3                         'led on
   pauseus 2                            'led on for 2 more usec
   low portb.3                          'led off
   pauseus 13                           'off time for led
   next                                 'repeat to complete waveform burst
   low portb.1                          'blue led ON

sense:                                  'check for objects in front, right, left
   if (lefteye = 0) AND (righteye = 0) then backup
   for n= 1 to 250                      'start burst 38 kc
   high portb.3                         'led on
   pauseus 2                            'led on for 2 more usec
   low portb.3                          'led off
   pauseus 13                           'off time for led
   next                                 'repeat to complete waveform burst
   if righteye = 0 then advoidleft      'detect object to the right
   for n= 1 to 250                      'start burst 38 kc
   high portb.3                         'led on
   pauseus 2                            'led on for 2 more usec
   low portb.3                          'led off
   pauseus 13                           'off time for led
   next                                 'repeat to complete waveform burst
   if lefteye = 0 then advoidright      'detect object to the left


forward:                                'move ahead
      high porta.1                      'pwr up servo
      for n=1 to 20                     'generate PWM for servos to move forward
      low porta.0                       'preset porta.0 to low
      pulsout porta.0,(lwheel+speed)    'left servo PWM high time
      low portb.5                       'preset portb.5 to low
      pulsout portb.5,(rwheel-speed)    'right servo PWM high time
      pause 20        'PWM low time
      if righteye = 0 then goto command 'look for IR from Remote
      next                              'repeat PWM for a bit of motion
      low portb.6                       'red led off
      f=f+1                             'increment forward motion counter
      if f>40 then goto scan            'clear forward path has been taken
      goto pulseme                      'restart another burst of IR
```

```
backup:                                 'backup
    f=0                                 'clear forward counts
    gosub zero                          'stop all servos
    gosub beep                          'beep
    high porta.1                        'pwr up servo
    for n=1 to 15                       'generate PWM for servos to backup
    low porta.0                         'preset porta.0 to low
    pulsout porta.0,(lwheel-(speed-5))  'left servo PWM high time
    low portb.5                         'preset portb.5 to low
    pulsout portb.5,(rwheel+(speed-5))  'right servo PWM high time
    pause 20                            'PWM low time
    next                                'repeat PWM for a bit of motion
    s=s+1                               'count backups
    if s>8 then goto turnright          'if >8 backups occur, then turn
    goto pulseme

advoidright:

    gosub avoidbackup

turnright:                              'turn right
    f=0                                 'clear forward counts
    s=5                                 'advance backup counts by 5 when turning
    high porta.1                        'pwr up servo
    for n=1 to 20                       'generate PWM for servos to turn right
    low porta.0                         'preset porta.0 to low
    pulsout porta.0,(lwheel+speed)      'left servo PWM high time
    low portb.5                         'preset portb.5 to low
    pulsout portb.5,(rwheel+speed)      'right servo PWM high time
    pause 20                            'PWM low time
    next                                'repeat PWM for a bit of motion
    goto pulseme                        'restart another burst of IR

advoidleft:

    gosub avoidbackup                   'backup a little before turning

turnleft:                               'turn left
    f=0                                 'clear forward counts
    s=2                                 'advance backup counts by 2 when turning
    high porta.1                        'pwr up servo
    for n=1 to 20                       'generate PWM for servos to turn left
    low porta.0                         'preset porta.0 to low
    pulsout porta.0,(lwheel-speed)      'left servo PWM high time
    low portb.5                         'preset portb.5 to low
```

```
        pulsout portb.5,(rwheel-speed)          'right servo PWM high time
        pause 20                                'PWM low time
        next                                    'repeat PWM for a bit of motion
        goto pulseme                            'restart another burst of IR

zero:                                           'stop the servos
        for n=1 to 5                            'generate PWM for servos to stop
        low porta.0                             'preset porta.0 to low
        pulsout porta.0,lwheel                  'left servo PWM high time
        low portb.5                             'preset portb.5 to low
        pulsout portb.5,rwheel                  'right servo PWM high time
        pause 20                                'PWM low time
        next                                    'repeat PWM to stop motion
        low porta.1                             'pwr down servo
        return

beep:
        sound portb.7, [75,10,110,10]           'make a sound
        return

scan:                                           'seeks ir beacon
        y=0                                     'clear scan turn count
        s=0                                     'clear backup counts

scan2:
        gosub zero
        f=0                                     'forward step counts
        y=y+1                                   'scan count
        high portb.1                            'turn on blue led
        high porta.3                            'beeper on
        serin porta.4,t1200,100,break,["a"],beakin      'receive beacon signal
        if (beakin >64) or (beakin <91) then beacon     'if valid character then goto beacon

break:                                          'continue rotation if no beacon found
        low portb.1                             'blue led off
        low porta.3                             'beeper off
        high porta.1                            'pwr up servo
        for n=1 to 8                            'scan right
        pulsout porta.0,(lwheel+speed)          'left servo PWM high time
        low portb.5                             'preset portb.5 to low
        pulsout portb.5,(rwheel+speed)          'right servo PWM high time
        pause 20                                'PWM low time
        next                                    'repeat PWM to rotate
        if y > 25 then hotbody                  'after turning enough degrees, clear y
        goto scan2                              'keep turning cw
```

```
beacon:
        low porta.3                              'turn off beeper
        sound portb.7, [80,100,120,300]          'make different sound
        goto forward                             'go toward beacon

hotbody:                                         'seeks heat
         y=0                                     'clear number of scan turns

hotbody2:                                        'continue with seeking heat
        gosub zero                               'stop motion
        y=y+1                                    'scan count
        pause 3000                               'wait for pyro to settle
        high portb.1                             'turn on blue led
        pause 100                                'keep blue led on
        low portb.1                              'turn off blue led
        for n=1 to 500                           'loop 500 seeks for heat
        if pyro = 1 then high portb.6            'turn on red led
        if pyro = 1 then rest = 0                'reset rest var to zero
        if pyro = 1 then goto pulseme            'goto pulseme (body of program)
        if irrec = 0 then goto beacon            'still can seek ir from operator
        low portb.1                              'turn off blue led
        pause 10                                 'if no heat, then loop every 10 ms
        next                                     'continue loop
        high porta.1                             'pwr up servo
        for n=1 to 30                            'start scan left
        pulsout porta.0,(lwheel-speed)           'left servo PWM high time
        low portb.5                              'preset portb.5 to low
        pulsout portb.5,(rwheel-speed)           'right servo PWM high time
        pause 20                                 'PWM low time
        next                                     'repeat PWM to rotate
        rest = rest + 1                          'increment rest value
        if rest > 18 then watchdog               'goto watchdog and wait
        if y > 7 then goto pulseme               'after turning around goto pulseme
        goto hotbody2                            'if less than 8 turns then turn cw

command:                                         'operator can make it turn to the right
        gosub beep                               'make a beep to alert operator signal recieved
        pause 500
        goto turnright                           'make a right turn
```

```
watchdog:
        low porta.1                         'turn off motor pwr
        low portb.1                         'turn blue led off
        high porta.3                        'start up a beep
        pause 2000                          'continue beep
        low porta.3                         'stop beep
        s=s+1                               'count watchdog disturbance
        if s > 4 then flash                 'rollie comes out if disturbed 5 times

watchdog2:

        if pyro = 1 then watchdog           'look for heat
        if push = 0 then watchdog3          'look for silent watchdog mode
        if righteye = 0 then flash          'look for IR-remote signal
        goto watchdog2

watchdog3:                                  'silent watchdog

        if righteye = 0 then flash          'look for IR-remote signal
        if pyro = 0 then watchdog3          'look for heat
        high portb.1                        'blue led turns on
        pause 1000                          'if it sees heat
        low portb.1                         'turn off blue led
        goto watchdog3                      'loop

avoidbackup:

        for n=1 to 15                       'generate PWM for servos to backup
        low porta.0                         'preset porta.0 to low
        pulsout porta.0,(lwheel-(speed-5))  'left servo PWM high time
        low portb.5                         'preset portb.5 to low
        pulsout portb.5,(rwheel+(speed-5))  'right servo PWM high time
        pause 20                            'PWM low time
        next                                'repeat PWM to backup a bit
        return

end
```