University of Florida

Department of Electrical and Computer Engineering

EEL 5666
Intelligent Machines Design Laboratory

Toolbot
Final Report

Final report
By
Jeno Nagy
December 10, 2002

Table of Contents

Abstract

Toolbot is an autonomous tool delivery system, which follows its user and dispenses sockets as requested by its user. The Toolbot has a small carousel with a handful of sockets that it can deliver. Using a combination of IR sensors, voice recognition, and a vision system, the Toolbot will provide assistance to a mechanic who needs tools for work beneath an automobile or other situations where a stationary toolbox in not appropriate or hard to get to.

Executive Summary

The Toolbot is a small round two-wheeled autonamous agent, which delivers sockets to its user. This is accomplished using 2 IR sensors for proximity detection, a CMUcam vision system for people following, a Voice Direct 364 voice recognition processor for voice commands through an RF link, specifically the names of the sockets needed, and various other supporting electronics such as beam breaks to get the task done.

Toolbot's electronics are controlled by a central microcontroller with various supporting devices attached. The controller is a PIC 16F877. This controller directs all behaviors including motion with two hacked servos. The carousel is aligned using a beam break for proper dispensing. Onboard supporting devices include a serial LCD driver IC, resistor networks for bump switches, and one RF receiver.

Off board devices consist of an RF headset with the Voice Direct 364 board in it. This is connected to another small PIC for serial transmission of voice commands to the Toolbot.

The robot's behaviors are dynamically affected by its environment, hence it is a programmed machine and autonomous. After startup the robot searches for the object to track in a circular motion. If it does not find the object within a specific given time, it wonders around randomly. It stops and checks again and again for the object to track. Once it locks in, it approaches it, and stops and waits to dispense a socket.

Introduction

Mechanics, or people like me, often find themselves underneath an automobile working on resolving a problem, or making modifications. A common job requires many different tools, which are often many feet away from where the mechanic needs it. It would be very convenient to have an aid dispense the proper sized tool as needed, when needed, to the mechanic. The space beneath a car in a common household garage, without lifts can be tight, and so is visibility to check whether the proper size wrench or socket is at hand. A small autonomous robot which, could dispense the proper tool with voice commands, would save time and the inconvenience (back-pain) of getting out from under the vehicle to get the right socket. This system would also keep the garage more organized and clutter free, and keep the mechanic from having to rearrange tools laying all over the floor after the job is done.

This paper covers all the major components of the robot, including the mobile platform design, actuation, sensors, the testing and structure of each component, and the behavior of the overall system with its surroundings.

Integrated System

The system is controlled by a PIC 16F877 microcontroller. Each task of the robot can be related to a specific part of the overall system. The movement of the platform by the driving servo system, behaviors of motion by the IR ranging system, bump ring, and CMUcam vision systems, and the socket dispensing task to the carousel structure and the voice command recognition system.

The theory of operation includes a user of the Toolbot. This user has a unique colored object or clothing at about eye level of the robot (so orange Gator pants or something similar). On power-up, the user stands in front of the camera at close proximity so that the unique color can be identified and tracked by the onboard vision system. Once the color is stored, the system checks for the color by spinning until it find its. If found, it follows it, else it randomly wonders around and checks periodically the same way. The flow chart of the operation can be seen on the next page, on Figure 1. Each independent sub-system on the robot is responsible for a specific task on the flow chart.

Initialization includes power-up, carousel alignment, and sub-systems check. Startup is a delay for the color lock to engage and the robot's readiness to follow an object.

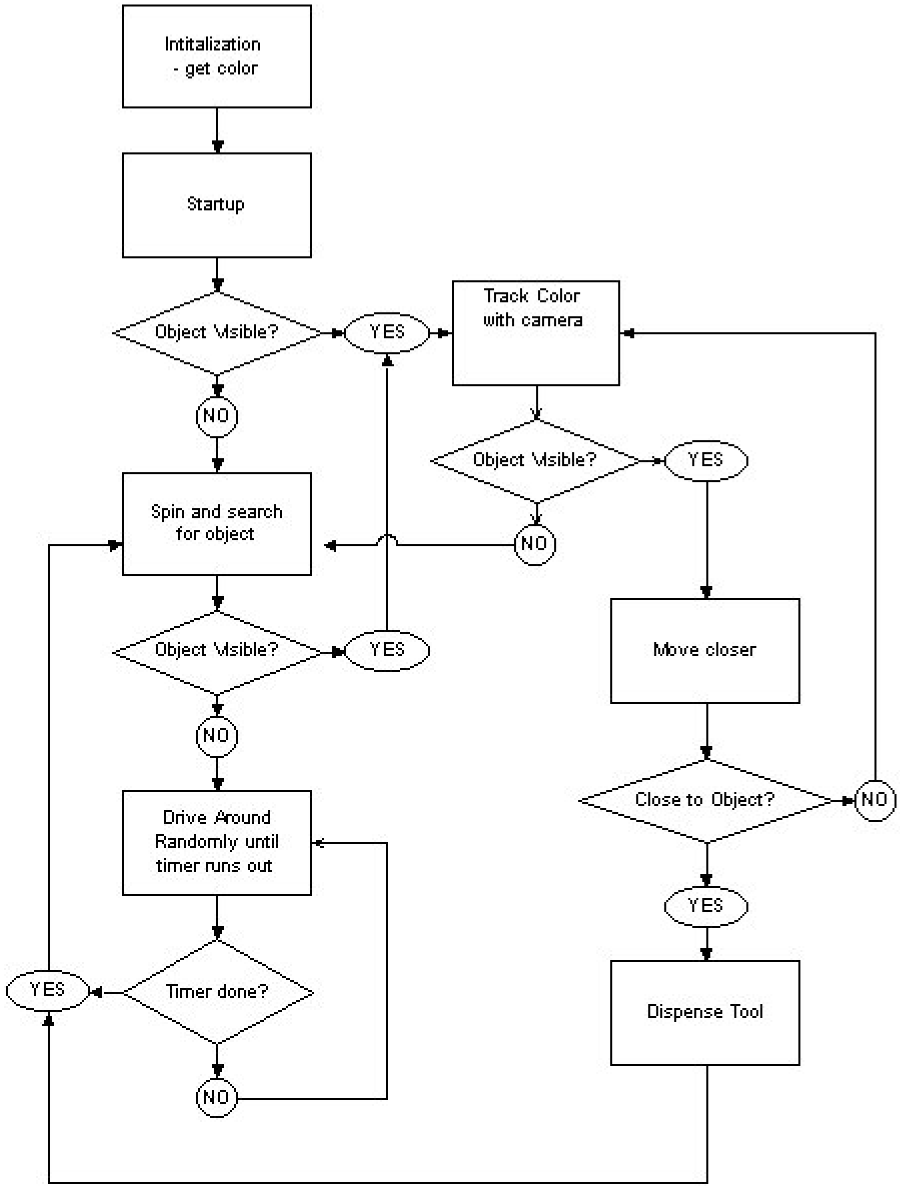

Figure 1. Theory of Operation

The code for the PIC16F877 is written in PIC Basic Pro. The structure of the code and the program flow can also be represented by the illustration above.

Mobile Platform

The platform is a small, seven inch diameter round platform with an overall height of approximately seven inches with the camera on top. The objectives for the platform include a small size to fit beneath automobiles and stability to carry ten sockets on top without tipping over. There is also a small sliding caster for balance in the back.

The main platform is shown below in figure 2 with cutouts for the two driving servo holders and front mount IR sensors.

Figure 2. Platform main base

The battery tray is located on the center section and is designed to hold a long, flat lead-acid YUASA NP2-12 battery. Two vertical supports hold up the second layer of the platform, which serves as the base for the carousel. This is approximately a six inch diameter circle, above which the same size carousel sits and spins. These pieces are shown on the following figures.

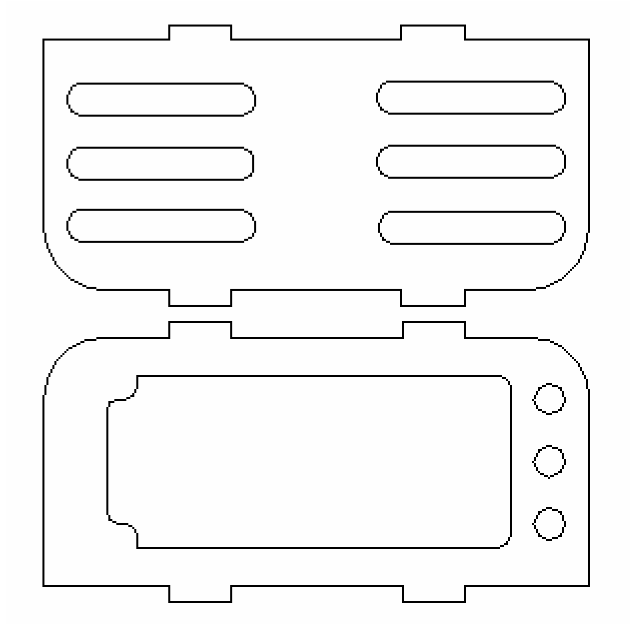

Figure 3. Vertical supports for second level

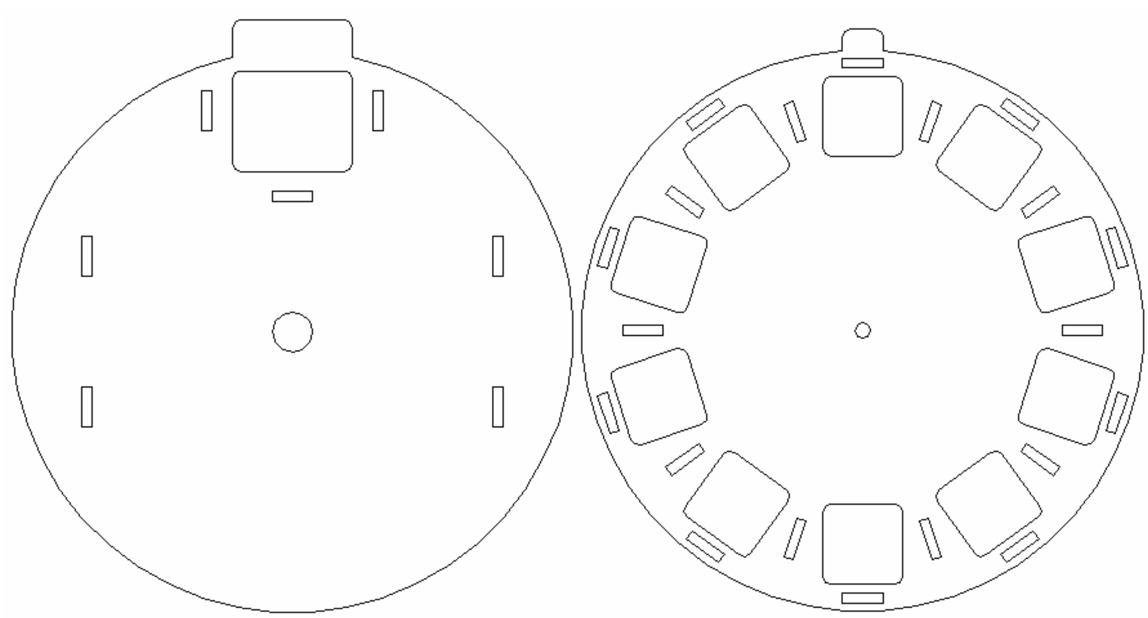

Figure 4. Second layer support and carousel

Regular sized sockets fit into the opening in the carousel. As it spins, the different sockets line up above the trap door opening on the second level base. The electronics fit between the two vertical supports and below the second layer. This keeps the electronic components protected and insulated from the moving parts above. Above the carousel, the third layer is supported by aluminum spacers. This serves as a top for the carousel as well as a base for the camera mount on top. The design of the third layer also has an opening up front to return the socket to the carousel.

Actuation

Actuation of the Toolbot consists of three different systems. The main actuation system is the two hacked ball-bearing servos used for motion of the platform. These are the GWS S03N BB pre-hacked servos, which produce 47 in-oz of torque.. The servos offer continuous rotation for motion and are controlled by PWM's form the PIC microcontroller. The servos require a 50Hz signal, hence every 20ms. The duty cycle determines the direction of the servo's rotation. The PIC's hardware PWM system is unable to generate such a low frequency signal, so software PWM was generated for driving signals.

The second actuation system is the servo, which controls the rotation carousel. This is also a GWS S03N BB servo, which is mounted at the center of the carousel. The PWM is again generated in software to rotate the carousel slowly.

The final actuation system is for the movement of the trap door to dispense the sockets. This is done with an unhacked GWS PICO BB servo. By applying the right PWN, the servo arm moves 90 degrees up or down. This pushes or lowers the door as needed. The following diagram shows the circuit of the actuating mechanisms. The code to control the servos is listed in the appendix.

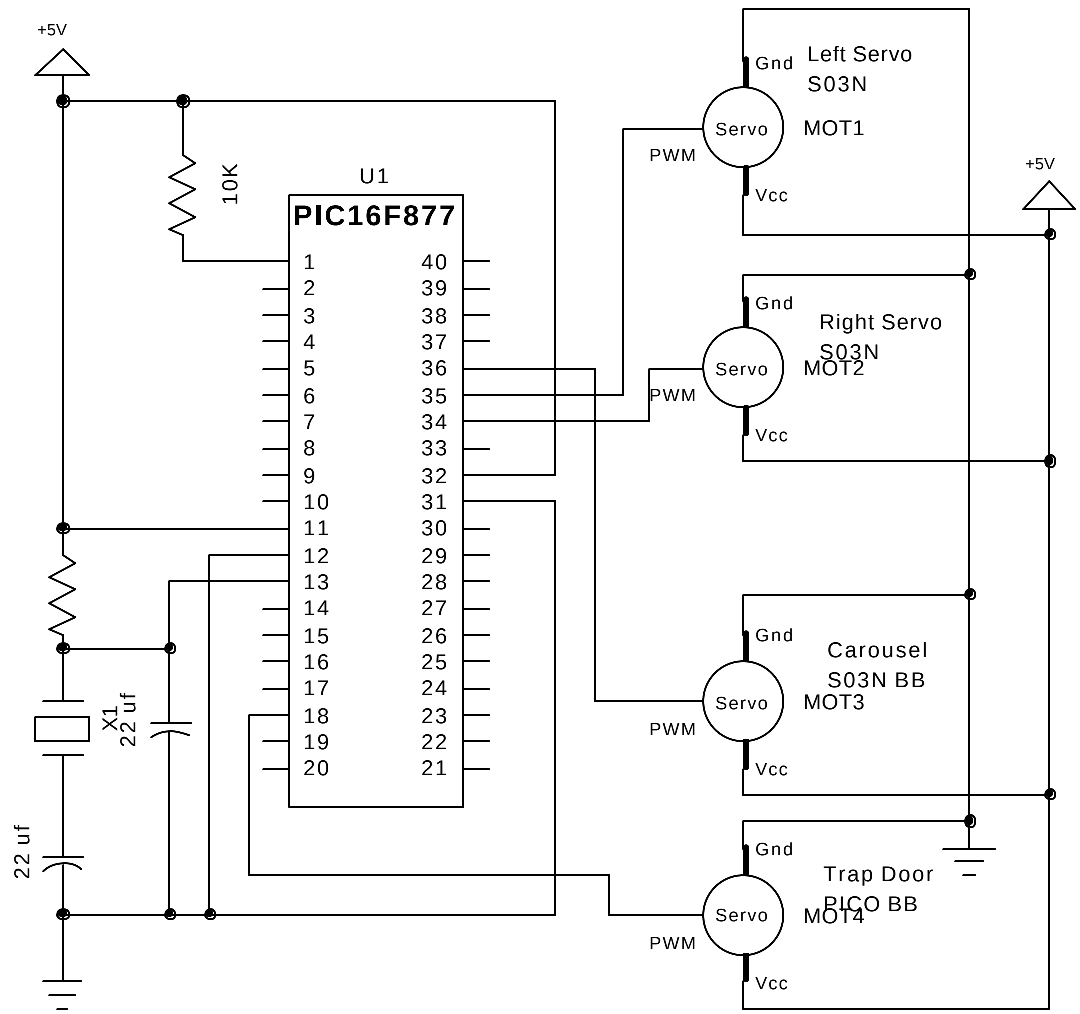

Figure 6. Actuation with various servos

One interesting thing I learned was the PIC's inability to use its hardware PWM to generate the 50Hz signal needed for proper PWM generation. The lowest resolution with the HPWM command using a 20Mhz oscillator is 1221Hz. With a 4 Mhz oscillator is 245Hz. The solution is to generate it in software, either trough the use of interrupts or some dynamic coding to ensure proper operation.

Sensors

Several sensors were used to achieve the mobility and goals set forth by the design objectives. The main sensors employed were IR proximity sensors, a beam break, bump ring system, voice recognition hardware, and the CMUcam for vision purposes. Each sensor is described below in more detail.

IR sensors

A pair of SHARP GP2D12 IR (Mark III robot store) proximity sensors were used on the front of the Toolbot to detect obstacles up ahead. The IR sensors are self modulated and demodulated, so only three connections are needed. Power, ground, and the analog signal coming out of each sensor to be read by the A/D system on the microcontroller. Using 8-bit resolution, the values range from 0 to 255. The useful range of these sensors allows positive detection up to a few inches from the front of the sensor. The max reading is about 130, which indicates an object about 4 inches in front. The performance of the sensors can be seen in the following graph of distance versus voltage output at the A/D port.

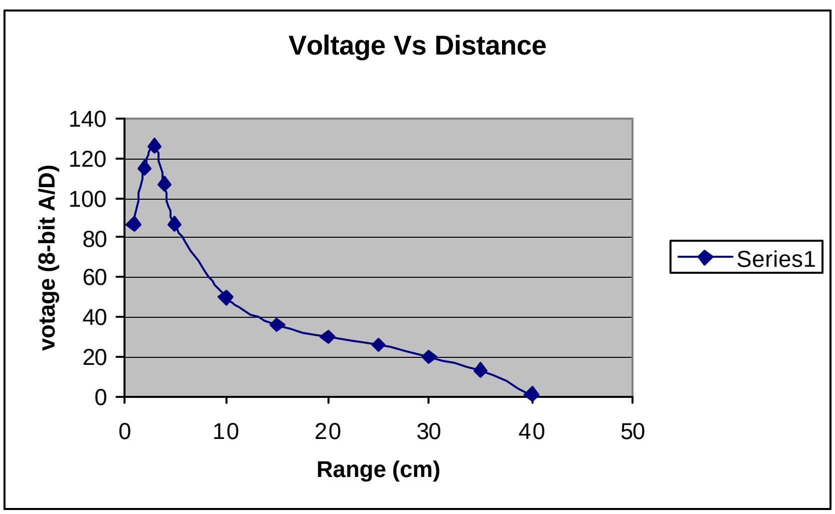

Figure 7. IR sensor output

The code to read the A/D ports and hence the IR sensors is located in the appendix.

Beam Break

A beam break is used to detect the alignment of the carousel with the first bin. This is crucial, since timing is needed to dispense the proper socket from the appropriate bin. The beam break is on Omron EE-SX3 emitter, detector pair (from Uriel Rodriquez). It is a self contained unit, with three inputs. Power, ground, and a signal line to detect open or closed beam status. The power must be connected trough a current limiting resitor, and for digital output, the output line should be pulled high. This gives a logical 1 when open, and 0 when closed. The theory of operation is simple. An IR LED diode emits light which a light sensistive transistor sees. When the beam is true, the output line is left floating, but when the beam is broken, the output line is pulled to ground. The circuit is shown below:

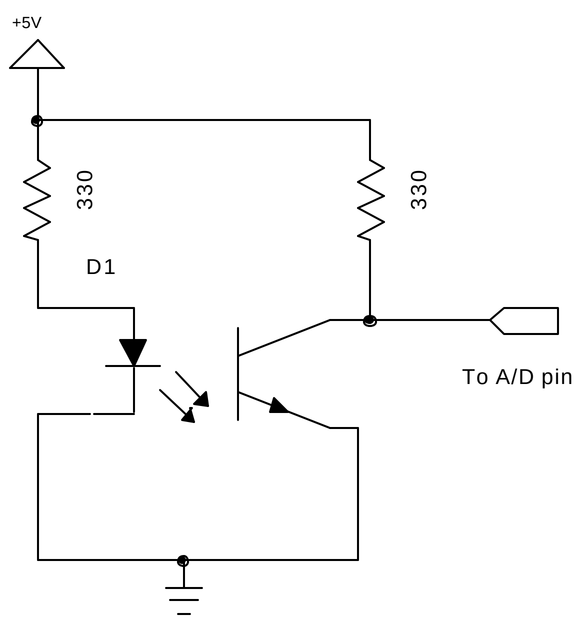

Figure 8. Beam Break

Bump Switch Network

The bump switch network is used to determine if a collision has occurred with Toolbot. If the IR sensors don't pick up an obstacle, the bump ring is the last effort to reverse direction or change course. The circuit design is from Uriel Rodriquez, and is implemented to save I/O pins on the controller. The bump network on Toolbot uses four contact switches to determine collisions and the direction they came form. The output of the network is tied to an A/D pin the PIC, from which the detection can be made. By using unique resistor values, a multiple voltage divider network can be made. This also works for multiple switches that close at the same time. The combination of resistors gives the voltage divider a unique value, which can be identified by the microcontroller for more precise movement in case of a collision. The circuit is shown below, and the code to detect it can be seen in the appendix.

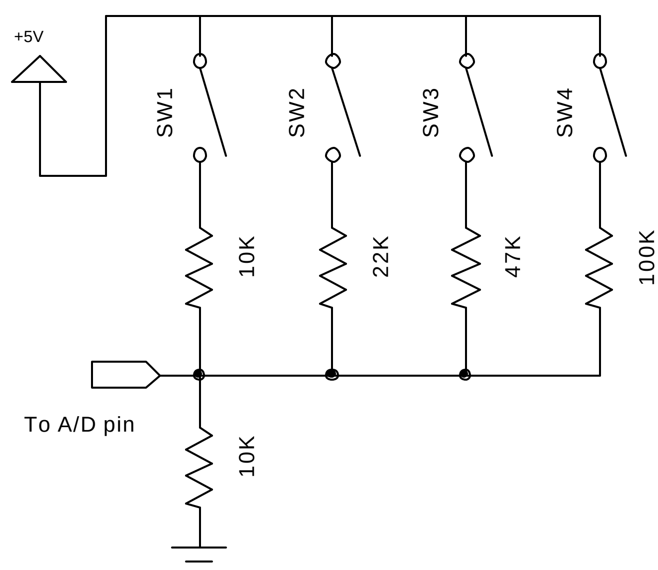

Figure 9. Bump switch network

Voice Recognition

Voice recognition is used on Toolbot to allow the user to select a particular size socket to be dispensed using only voice. Voice recognition is a difficult sensory project, but there are several manufactured sub-systems, which can accomplish this task. Toolbot uses the Sensory Voice Direct 364 (from Jameco) to recognize a spoken work and send a digital signal which represent the work said.

The Voice Direct Board is set up for strict, continuous listening mode, which allows the user to say one key work, followed by up to fifteen additional words. Once the works are trained, the system is ready to listen to commands. When the key word is recognized, the secondary word is listened to. If this is also recognized, certain pins will be toggled high for 1 second to indicate a match.

Two designs were tried, one on-board the robot with a miniature microphone, and second a remote, RF link, with a transmitter in a remote control box, and the receiver on-board Toolbot. The on-board idea did not work well, since voice tone and attenuation must be exact to the trained words, else a false or no recognition is made.

The RF link was a much improved idea. It employs the Voice Direct board, another small PIC 16F84 microcontroller, and the Rentron TWS-434 RF transmitter (from Reynolds Electronics) all enclosed in a small box, with a microphone headset. The digital data from the voice board is read by the PIC and is sent serially at 2400 baud to the receiver on Toolbot. The receiver is the matched pair RWS-434, which then serially transmits the data to the main controller on Toolbot. The range as tested indoors was effective up to 25 feet, but Rentron claims distances of 400 feet. The circuit for the transmitter and receiver are shown on the following page. The code for both are listed in the appendix.

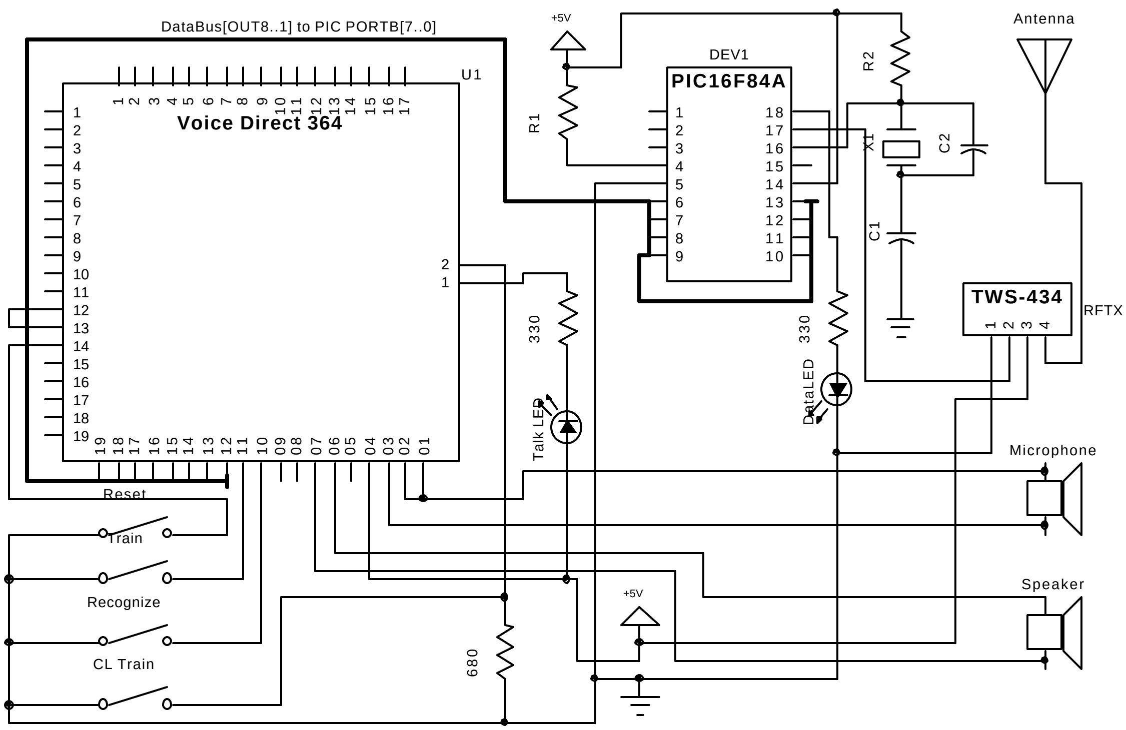

Figure 10. Voice Recognition RF transmitter headset

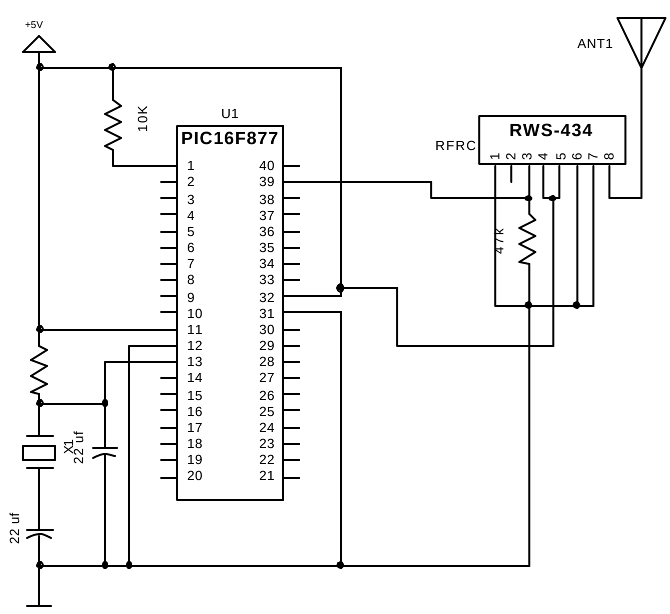

Figure 11. Voice recognition RF receiver

CMUcam Vision system

People following is implemented using the CMUcam serially connected to the main PIC controller. The CMUcam (From Seattle Robotics) is an integrated digital CMOS camera with an SX-28 microcontroller. Digital image information is extracted and serially transmitted. The theory of operation is that the camera can lock onto and track objects with bright colors. It does this by taking the RGB values of the tracked item, and following it on the screen. The camera then dumps serially image information, such as mass Y, mass X, and pixel information. This can be used to control the motion of Toolbot to keep the object centered in the camera's lens.

After many hours of terrible headaches and angry outburst of hatred, I got the camera to work and send serial data. (Despite having the wrong lens shipped form Seattle Robotics. NOTE: make sure you get the improved IR wide angle lens, its free if you ask). Initially the serial link was at 9600 and in pure bi-directional ASCII, but later improved to 38,400 baud with ASCII command transmission from PIC to camera, but raw stream data back form the camera to the PIC. This improved calculation time, since all values sent back were in standard 8-bit numerical format, as opposed to three 8-bit values representing the correct value ("2","5","1", for "251" for example).

The camera is wired to a pair of input and output pins for receiving and transmitting. For various speed setting, jumpers must be set on the camera board. For 38,400, set jumper 2 only. Toolbot uses TTL logic level serial transmission, which the CMUcam offers as separate output pins on the board. These pins are also tied internally to a Max233 level shifter IC. This needs to be removed from its socket, otherwise the TTL logic pins do not work.

The CMUcam is initialized in a sequence of commands sent to the camera. The following shows how to lock on to a color on front of the camera and track its center of mass, its pixel count, and the camera's confidence level at seeing the object.

```
SerOut2    CMUcamTX,6,["rs",13]     'Camera reset
SerIn2     CMUcamRC,6,[WAIT(":")]
SerOut2   CMUcamTX,6,["PM 1",13]'poll mode only
SerIn2     CMUcamRC,6,[WAIT(":")]
SerOut2    CMUcamTX,6,["MM 1",13]'mass mode on
SerIn2     CMUcamRC,6,[WAIT(":")]
SerOut2   CMUcamTX,6,["CR 18 32 19 32",13] 'turn auto gain, white balace off
SerIn2    CMUcamRC,6,[WAIT(":")
SerOut2    CMUcamTX,6,["RM 3",13]
SerOut2    CMUCamTX,6,["TW",13]
SerIn2    CMUcamRC,6,[WAIT(":")]
LOOP
'SerOut2         CMUcamTX,6,["TC",13]
SerIn2….DATA NEEDED
GOTO LOOP
```

The camera is reset on software, set to poll mode, mass mode is engaged and the image is adjusted by turning auto-gain and white balance off. Raw mode is turned on for raw transmission from camera to PIC. Then Track Window is called to lock in on object in the center part of the image. The RGB values of this object are tracked by calling the Track Color command. One frame of values is sent back each time. Toolbot checks to see if the center of mass in the X direction is close to the center. The screen size is 80 by 143, so the X center is at 40. If the mean is between 35 and 55, then move forward, else if les then 35 move left, or more that 55 move right. The pixel count indicated how many pixels the object takes up, hence the distance form it. If the pixel count > 208, then the object is close, so Toolbot stops. Confidence level indicates the camera's current lock on the color. If the confidence is low, the object is not in view.

Behaviors

Toolbot behaviors can be categorized into several subsections. The robot switches from behavior to behavior based on the dynamic changes in its surroundings. The behaviors are Familiarization, People Following with obstacle avoidance, Searching, Random Roaming with obstacle avoidance, and Tool Deposit. Each is described below.

Familiarization

On startup, toolbot must familiarize itself with the object it is requested to track. This is done using the CMUcam's Track Window command. Once the camera is warmed up and the proper modes are set, The camera takes a snap shot of what ever is in the center of its lens, and locks in on that RGB color combination. The familiarization takes a few seconds, with a green LED status light blinking to alert the user. This behavior state is only entered once, during powerup. If for some reason there are tracking issues, Toolbot must be turned off and reinitialized.

Searching

In order to achieve people following skills, some searching is involved to find the object being tracked. The searching algorithm is very simple, and this behavior is entered after familiarization, and then on certain time intervals of the object is not tracked or seen for a while. The camera takes an initial reading after familiarization, and Toolbot determines wether the object is seen. If it is, then Toolbot moves toward the object, if not, then the search behavior is initiated. Toolbot spins in place twice around its own axis, slowly, to see if the object can be seen in a 360 degree motion near by. If it is located, the search behavior is switched to people/object following. If not seen, then random roaming is initiated. This gives Toolbot a chance to move to another part of the world, and try searching again.

Random Roaming

Random roaming occurs when a search fails to provide definite direction for Toolbot to follow. In random roaming, toolbot moves forward after a spin search. It continues moving forward until obstacles are detected by either the IR sensors or the bumpswitch network. Then the appropriate action is taken. The actions can be seen from the flow chart below.

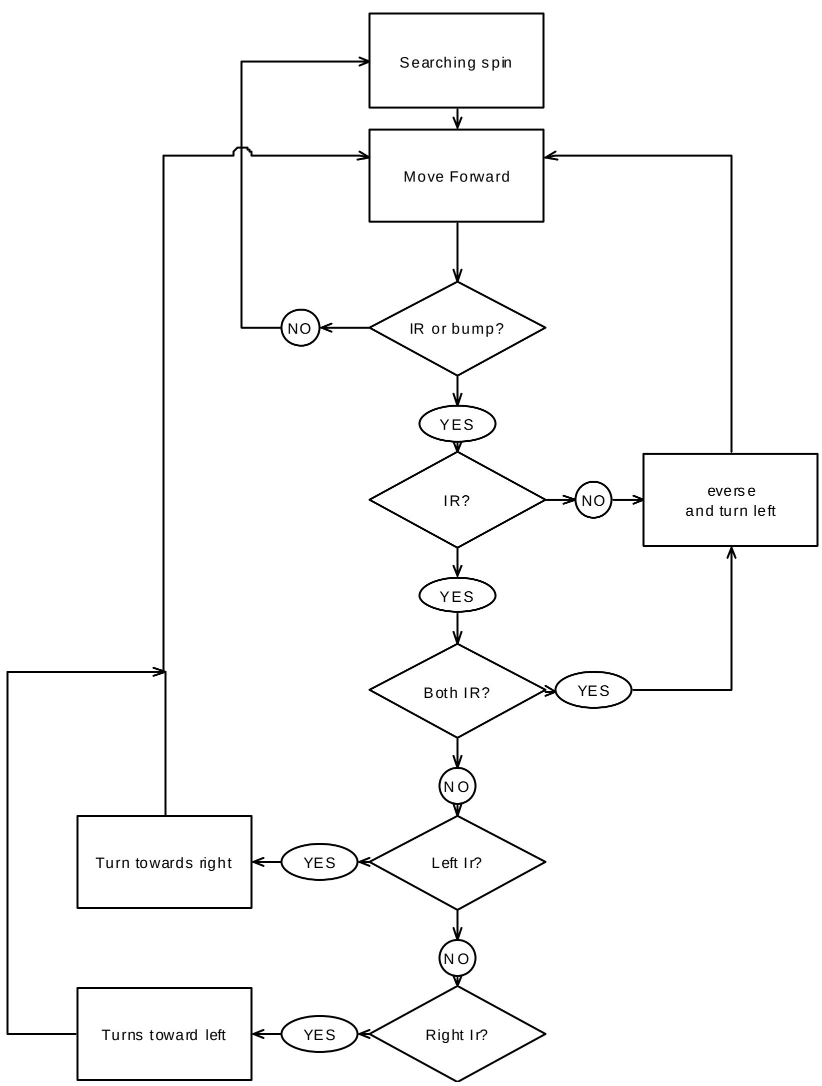

Figure 13. Random roaming behavior

People Following

People following (object following) is entered when the camera has a positive lock on the object it needs to track. Toolbot calculates its movement based on the location of the middle mass of the object, specifically in the X axis. Y axis calculations are ignored, but could be implemented in the future for up, down looking also. If the middle mass is within the range for forward motion, Toolbot moves forward for a small amount of time, and then reevaluates for its next move. If the object is on either extreme of the line of sight, then Toolbot turns slowly towards the object until it is near center again, then moves forward. If the object is lost at any time in this process, Toolbot stops and goes into search mode. The search mode is the same one as on initialization, with a spin about its own axis, unless the object is lost while it was in sight and Toolbot was turning. For example, if Toolbot is turning to the left, and then the object fall out of view, Toolbot assumes it is because the object was moving too fast towards the left and continuous t spin left. The same of true for the right side.

While moving forward, the same steps are checked for obstacles as in the random motion behavior, Bump sensors take priority over IR sensors, which over the camera. So even if there is a positive lock on the object to move forward, if there are obstacles ahead, Toolbot will turn appropriately, and if the object is lost, will go into search mode.

One major lesson learned again with the PIC's inability to generate the proper PWM from hardware. Without this, Toolbot must stop and revaluate its motion direction, so that the software can react and generate the PWM. With hardware, the motors would be running constantly and only updating the direction values would be needed, producing a much smother operation cycle.

Tool Deposit

Once Toolbot is close to its target object, it stops and offers a socket to its user. It signals by displaying the appropriate message on the LCD display, as well as flashing a red LED indicator. Toolbot waits for serial commands from the user. If no commands area sent, Toolbot rechecks the status of the camera every 5 seconds to see if the target has moved without needed tools. If still there, it keeps waiting, if moved it follows.

If a socket is needed, it is dispensed and Toolbot waits for the socket to be returned and the return button pressed. It then rechecks the camera status and enters one of the other behaviors as described earlier.

Experimental Layout and Results

The main code to operate Toolbot was tested in sections and once successful, they were combined to form the overall flow of behaviors described before. Tests were performed on each section to ensure proper operation. The major sections are listed below.

Basic Obstacle Avoidance

Since obstacle avoidance is required in all other moving behaviors it was the first one to be completed and tested. The IR sensor data can be found under the sensors section. The IR sensor data controls Toolbot with various IF..THEN..ELSE statements. The value of the IR sensors was adjusted until a comfortable turning distance was seen from the obstacle ahead. The results gave a nice turn and provided good obstacle avoidance.

Carousel Alignment

To dispense the proper socket, the carousel on Toolbot must stop above the trap door within some degree of accuracy. After several tests, I realized that even if the carousel is aligned initially, it losses alignment after dispensing a few sockets. This was solved by realigning the carousel to its initial setting each time a socket is returned. This keeps the timing close, and only glitches a few times.

Conclusion

Toolbot overall performs as needed by the original design. The only issue is with the camera and its inconsistency with lighting. Under most circumstances it performs excellent and finds its user, but there are times when it takes several tries before it gets a good lock on its object. I have another IR filter coming in the mail, which should fix the camera's susceptibility to external light.

Limitations of my work were mainly time related. I have more ideas to try to improve the design further, but time only allows so much. Several areas of the design worked better than I had expected. The voice recognition with the RF headset was nice compared to yelling at the microphone a few feet away. The platform design worked really well. It is balanced, compact, and fairly neat. Areas of improvement focus on the carousel's instability sometimes and again the CMUcam's issues.

Technical issues with the PIC leave me to say this: Don't use one if you need to run servos continuously in the background, unless some interrupt services can be set up to correct this problem. The Atmels did not have this problem. Also if anybody decides to use the CMUcam, give yourself plenty of time to experiment with it to learn it (and maybe buy two, so you can throw one against the wall). After some experimenting it does work well, with proper lighting.

If I would start the project over, I would of used an Atmel microcontroller for its better features. For future enhancements on Toolbot, I'd like to increase the payload to more tools, voice playback system, and possibly a servo activated camera mount so that it can swivel left right, up and down, for more degrees of tracking freedom.

Overall the project was fun, I learned a lot of new things, and things not to do.

Documentation

Credits

I'd like to give credit and thanks for all the help I was given by Dr. Arroyo, Dr. Swartz, Uriel Rodriquez, and Jason Plew in carving Toolbot out of a pile of resistors and two sheets of plywood.

Uriel provided me with several key parts and ideas for Toolbot, including the bump network system, and the break beam for the carousel.

There were several good websites, which I found useful in helping debug the subsystems of Toolbot. In no particular order:

Websites of relative importance

(PICs)
<http://www.planetmicrochip.com>
Good site for data sheets, same code, etc. for the PIC, after all they make them

<http://electronicKits.com>
Cheap PIC programmer works real good

(RF transmitter/receiver)
<http://www.rentron.com>
Lots of stuff on IR and RF decoding, encoding, and not too pricey

(CMUcam)
<http://www.seattlerobotics.com>
Origin of the camera, ask for the proper IR lens.

<htttp://www.acroname.com>
More CMUcam stuff, they sell it too, sample codes

(Servos in general)
<http://RobotStore.com>
Some useful basic info on servos and controls for them

Parts List

Most of the parts were purchased through the internet, but some items were provided by the TA's. Only the significant pieces are listed on the following table.

Part	Part#	Supplier	Contact
PIC 16F877	176882	Jameco	http://www.jameco.com
PICProto Board64	13652	Jameco	http://www.jameco.com
PICALL programmer	DIY KIT 117	Carl's Electronics	http://electronickits.com
Servo Wheels		Mark III	http://www.junun.org/MarkIII
Sharp GP2D12 IR		Mark III	http://www.junun.org/MarkIII
Servos		Mark III	http://www.junun.org/MarkIII
Voice Direct 364	173489	Jameco	http://www.jameco.com
Serial LCD driver	171951	Jameco	http://www.jameco.com
CMUcam		Seattle Robotics	http://www.seattlerobotics.com

Appendix

The appendix contains the final code in PicBasic Pro used to control Toolbot.

```
' This code controls Toolbot
' Jeno Nagy
' 12-9-02

DEFINE  OSC         20
DEFINE  ADC_BITS        8       ' Set number of bits in result
DEFINE  ADC_CLOCK       3       ' Set clock source (3=rc)
DEFINE  ADC_SAMPLEUS    50      ' Set sampling time in uS

MiddeX              CON     40
MiddleY             CON 70
LFMax               CON 1250        'Left servo forward max speed
LRMax               CON 250 'Right servo forward max speed
LStop               CON 0   'Left servo stop speed
RFMax               CON     250     'Right servo forward max speed
RRMax               CON     1250    'Right servo reverse max speed
RStop               CON     0       'Right servo stop speed
RefreshPeriod   CON 20  'refresh servo pins delay
CarSpeed            CON     825     'Carousel Speed

LEDArray        VAR PORTC.2
LMotor              VAR     PORTB.1         'Left servo
RMotor              VAR     PORTB.2         'Right servo
CMUcamRC            VAR     PORTC.0
CMUcamTX            VAR PORTC.1
LCD                         VAR PORTB.0
Door                VAR PORTC.3
Carousel        VAR PORTB.3
RFin                VAR PORTB.6
RedSwitch           VAR PORTB.5
RedLED              VAR PORTB.4

Bumper              VAR BYTE                'A/D bumper switch
LeftIR              VAR BYTE                'A/D left IR
RightIR             VAR BYTE
```

```
BBreak          VAR BYTE
DoorCounter     VAR BYTE
SpinDelay       VAR BYTE
BinNumber       VAR BYTE
Nothing         VAR BYTE
Pattern         VAR BYTE
IdleCount       VAR BYTE
MX              VAR BYTE
MY              VAR BYTE
Confidence      VAR BYTE
Pixels          VAR BYTE
RawMode         VAR BYTE[9]
LSpeed          VAR WORD
RSpeed          VAR WORD
LastDirection   VAR BYTE
Counter         VAR BYTE
BackCount       VAR BYTE

    Pause   50
    TRISA = %11111111       ' Set PORTA to all input
    ADCON1 = %00000010      ' Set PORTA analog
    Output  LCD
    Output  LEDArray
    Output  LMotor
    Output  RMotor
    Input   RedSwitch
    Input   CMUcamRC
    Low     LMotor
    Low     RMotor
    Low     LCD
    LastDirection = 2
    DoorCounter = 0
    SpinDelay = 0
    BinNumber = 1
    Nothing = 0
```

```
        BackCount = 0
        Counter = 0
        Pattern = 0
        IdleCount = 0
        Pause     1000
        SerOut    LCD,2,[$FE,$01]
        LSpeed = LStop
        RSpeed = RStop
        GoSub     BlinkLEDArray
        GoSub     BlinkLEDArray
        GoSub     BlinkRedLED
        GoSub     BlinkRedLED
InitDoor:
        While (DoorCounter < 50)
                PulsOut   Door,250
                Pause     RefreshPeriod
                DoorCounter = DoorCounter + 1
        Wend
        Pause 500
        DoorCounter = 0
        While (DoorCounter < 50)
                PulsOut   Door,1250
                Pause     RefreshPeriod
                DoorCounter = DoorCounter + 1
        Wend
        DoorCounter = 0
InitCarousel:
        ADCIN     3,BBreak
        Pause     10
        IF (BBreak > 100) Then
                PulsOut   Carousel,CarSpeed
                Pause     RefreshPeriod
        Else
                GoTo      InitCMUcam
        EndIF
```

```
        GoTo    InitCarousel
InitCMUcam:
        GoSub   BlinkLEDArray
        Pause   3000                ' Wait 5 second for image stabilization
        SerOut  LCD,2,[$FE,$01]
        Pause   50
        SerOut2 CMUcamTX,6,["rs",13]        'Camera reset
        SerIn2  CMUcamRC,6,[WAIT(":")]
        Pause   1
        GoSub   BlinkLEDArray
        GoSub   blinkCMU
        SerOut2 CMUcamTX,6,["PM 1",13]      'poll mode only
        SerIn2  CMUcamRC,6,[WAIT(":")]
        Pause   1
        GoSub   blinkCMU
        SerOut2 CMUcamTX,6,["MM 1",13]      'mass mode on
        SerIn2  CMUcamRC,6,[WAIT(":")]
        Pause   1
        GoSub   blinkCMU
        SerOut2 CMUcamTX,6,["CR 18 32 19 32",13] 'turn auto gain, white balace off
        SerIn2  CMUcamRC,6,[WAIT(":")]
        Pause   1
        GoSub   blinkCMU
        SerOut2 CMUcamTX,6,["RM 3",13]
        Pause   1
        GoSub   blinkCMU
        SerOut2 CMUCamTX,6,["TW",13]
        SerIn2  CMUcamRC,6,[WAIT(":")]
        Pause   1
        GoSub   BlinkCMU
        SerOut2 LCD,84,["Tracking Mass"]
        Pause 1
Track:
        Pixels = 0
        Confidence = 0
```

```
IF (LastDirection = 4) Then
        LastDirection = 0
        LSPeed = LStop
        RSpeed = RStop
        Pause   500
EndIF
GoSub   CAMTrack
IF (Confidence < 8) Then
        Low     LEDArray
                LSpeed = LFMax
                RSpeed = RRMax
                IdleCount = IdleCount + 1
                IF IdleCount > 80 Then
                        IdleCount = 0
                        GoTo DriveAround
                EndIF
        IF (LastDirection = 2) OR (LastDirection = 4) Then
                IF (BackCount < 2) Then
                        LSpeed = LRMax + 200
                        RSpeed = RRMax - 200
                        BackCount = BackCount + 1
                Else
                        LSpeed = LFMax
                        RSpeed = RFMax
                        BackCount = 0
                        LastDirection = 0
                EndIF
        Else
                IF (LastDirection = 1) Then
                        LSpeed = LRMax
                        RSpeed = RFMax
                Else
                        IF (LastDirection = 3) Then
                                LSpeed = LFmax
                                RSpeed = RFmax
```

```
                EndIF
            EndIF
        EndIF
Else
        High    LEDArray
        IF      (Pixels > 20) AND (Pixels < 207) Then
            IF (MX < 35) Then
                LSpeed = LRMax
                RSpeed = RFMax
                LastDirection = 1
                IdleCount = 0
            Else
                IF (MX > 45) Then
                    LSpeed = LFMax
                    RSpeed = RRMax
                    LastDirection = 3
                    IdleCount = 0
                Else
                    LSpeed = LFMax
                    RSpeed = RFMax
                    LastDirection = 2
                    IdleCount = 0
                EndIF
            EndIF
        Else
            IF (Pixels > 208) Then
                LastDirection = 4
                IdleCount = 0
                GoTo    Dispense
            EndIF
        EndIF
EndIF
IF (LastDirection = 2) Then
        While (Counter < 40)
            PulsOut   LMotor,LSpeed
```

```
                PulsOut RMotor,RSpeed
                Pause      RefreshPeriod
                Counter = Counter + 1
        Wend
        Counter = 0
    Else
        PulsOut    LMotor,LSpeed
        PulsOut RMotor,RSpeed
    EndIF
    GoTo       Track


CAMTrack:
    SerOut2   CMUcamTX,6,["TC",13]
    SerIn2    CMUcamRC,6,[WAIT("M"),STR Rawmode\8]
    MX = Rawmode(0)
    MY = Rawmode(1)
    Pixels = RawMode(6)
    Confidence = RawMode(7)
    'SerOut2  LCD,84,["X:",DEC MX," Y:",DEC MY]
    'SerOut    LCD,2,[$FE,$C0]
    'SerOut2 LCD,84,["P:",DEC Pixels," C:",DEC Confidence]
    'Pause 500
    'SerOut2  LCD,84[$FE,$01]
    Pause     1
    Return

Dispense:
    Pattern = 0
        DoorCounter = 0
        SpinDelay = 0
        BinNumber = 1
    GoSub     BlinkRedLED
    ADCIN     0, LeftIR
    ADCIN     1, RightIR
```

```
Pause	10
IF (LeftIR > 60) OR (RightIR > 60) Then
	High	RedLED
	SerOut	LCD,2,[$FE,$01]
	Pause	1000
	SerOut2	LCD,84,["Need a socket?"]
	High	LEDArray
	SerIn	Rfin,4,10000,GoTrack,["MESSAGE"],Pattern
	Low		LEDArray
	IF (Pattern < 6) OR (Pattern > 15) Then
		GoTo	Dispense
	EndIF
	SerOut	LCD,2,[$FE,$01]
	Pause	1000
	SerOut2	LCD,84,["Get Tool#:",DEC Pattern,"mm"]
	While (BinNumber <> (Pattern - 5))
		While (SpinDelay < 7)
			PulsOut	Carousel,CarSpeed
			Pause	RefreshPeriod
			SpinDelay = SpinDelay + 1
		Wend
		SpinDelay = 0
		BinNumber = BinNumber + 1
		IF BinNumber = 11 Then
			BinNumber = 1
		EndIF
	Wend
SpinDelay = 0
IF (Pattern > 10) Then
	While (SpinDelay < 2)
		PulsOut	Carousel,CarSpeed
		Pause	RefreshPeriod
		SpinDelay = SpinDelay + 1
	Wend
EndIF
```

```
            While (DoorCounter < 50)
                        PulsOut   Door,250
                        Pause     RefreshPeriod
                        DoorCounter = DoorCounter + 1
            Wend
            Pause 2000
            DoorCounter = 0
            While (DoorCounter < 50)
                        PulsOut   Door,1250
                        Pause     RefreshPeriod
                        DoorCounter = DoorCounter + 1
            Wend
            SerOut    LCD,2,[$FE,$01]
            Pause     1000
            SerOut2   LCD,84,["Return socket and"]
            SerOut LCD,2,[$FE,$C0]
            SerOut2   LCD,84,["press red button."]
      WaitforReturn:
            IF (RedSwitch <> 1) Then
                        GoSub     BlinkRedLED
                        GoTo      WaitForReturn
            EndIF
      EndIF
GoTrack:
      SerOut    LCD,2,[$FE,$01]
      Low                 RedLED
      GoTo      Track

DriveAround:


      GoSub     BlinkRedLED
      GoSub     BlinkLEDArray
      SerOut    LCD,2,[$FE,$01]
      Pause     1000
      SerOut2   LCD,84,["Anybody here"]
```

```
        SerOut LCD,2,[$FE,$C0]
        SerOut2  LCD,84,["needs sockets?"]
                ADCIN   0,RightIR
                ADCIN   1,LEFTIR
                ADCIN   2,Bumper
                LSpeed = LStop
                RSpeed = RStop
                Pause   RefreshPeriod
                IdleCount = 0
CheckSensors:
                Counter = 0
                IF      (RightIR < 70) AND (LeftIR < 70) AND (Bumper < 50) Then
                        LSpeed = LFmax
                        RSpeed = RFMax
                        PulsOut   LMotor,LSpeed
                        PulsOut  RMotor,RSpeed
                        Pause    RefreshPeriod
                Else
                        IF (RightIR < 70) AND (LeftIR > 70) AND (Bumper < 50) Then
                                LSpeed = LFMax
                                While (Counter < 25)
                                        RSpeed = RSpeed + 4
                                        PulsOut   LMotor,LSpeed
                                        PulsOut  RMotor,RSpeed
                                        Pause    RefreshPeriod
                                        Counter = Counter + 1
                                Wend
                                Counter = 0
                        Else
                                IF (RightIR > 70) AND (LeftIR < 70) AND (Bumper < 50) Then
                                        RSpeed = RFMax
                                        While (Counter < 25)
                                                LSpeed = LSpeed - 4
                                                PulsOut LMotor,LSpeed
                                                PulsOut  RMotor,RSpeed
```

```
                Pause    RefreshPeriod
                Counter = Counter + 1
        Wend
        Counter = 0
Else
        IF (RightIR > 70) AND (LeftIR > 70) AND (Bumper < 50) Then
                While (LSpeed > LRMax) AND (RSpeed < RRMax)
                        LSpeed = LSpeed - 4
                        RSpeed = RSpeed + 4
                        PulsOut LMotor,LSpeed
                        PulsOut RMotor,RSpeed
                        Pause    RefreshPeriod
                Wend
                While (Counter < 50)
                        PulsOut LMotor,LSpeed
                        PulsOut RMotor,RSpeed
                        Pause    RefreshPeriod
                        Counter = Counter + 1
                Wend
                Counter = 0
                LSpeed = LRmax
                RSpeed = RFmax
                While (Counter < 15)
                        PulsOut LMotor,LSpeed
                        PulsOut RMotor,RSpeed
                        Pause    RefreshPeriod
                        Counter = Counter + 1
                Wend
                Counter = 0
        Else
                IF (Bumper > 50) Then
                        LSpeed = LStop
                        RSpeed = RStop
                        PulsOut LMotor,LSpeed
                        PulsOut RMotor,RSpeed
```

```
                                        Pause     RefreshPeriod
                                        LSpeed = LRMax
                                        RSpeed = RRMax
                                        While (Counter < 70)
                                                PulsOut LMotor,LSpeed
                                                PulsOut RMotor,RSpeed
                                                Pause     RefreshPeriod
                                                Counter = Counter + 1
                                        Wend
                                        Counter = 0
                                        LSpeed = LRmax
                                        RSpeed = RFmax
                                        While (Counter < 30)
                                                PulsOut LMotor,LSpeed
                                                PulsOut RMotor,RSpeed
                                                Pause     RefreshPeriod
                                                Counter = Counter + 1
                                        Wend
                                        Counter = 0
                                EndIF
                        EndIF
                EndIF
        EndIF
EndIF
PulsOut   LMotor,LSpeed
PulsOut RMotor,RSpeed
Pause     RefreshPeriod
ADCIN     0,RightIR
ADCIN     1,LeftIR
ADCIN     2,Bumper
IdleCount = IdleCount + 1
IF (IdleCount > 200) Then
        IdleCount = 0
        Nothing = 0
        GoTo      Track
```

```
        EndIF
        GoTo    CheckSensors
    End
    Return

BlinkCMU:
    SerOut2 CMUcamTX,6,["L1 1",13]
    Pause   250
    SerOut2 CMUcamTX,6,["L1 0",13]
    Pause   250
    Return

BlinkLEDArray:
    High    LEDArray
    Pause   200
    Low             LEDArray
    Pause   200
    Return

BlinkRedLED:
    High    RedLED
    Pause   200
    Low             RedLED
    Pause   200
    Return

    End
```