University of Florida
Department of Electrical and Computer Engineering
EEL5666: Intelligent Machine Design Laboratory

# WALLY
## An Autonomous Inventory Retrieval System

Final Report

Kenneth Rosenthal II
Dr. A. Antonio Arroyo
April 25, 2006

# Table of Contents

## Abstract

I will build a robot that will be able to find and retrieve specific items from a warehouse. The robot will allow a user to enter the item they wish to retrieve. Once the robot receives this information it will set out for the item by following a black line through a model warehouse. It will avoid collisions and be cautious of detecting unseen collisions. It will then scan rooms by using a RFID reader. Once it finds the room which houses the sought after item, it will enter the room, pick up the item and take it to a specific drop point. Once it completes this task it will return to its start position and await the next command. I hope to also allow the robot to move items from one room to another. My goal is to create a completely autonomous method for a company to fill, pack, and ship orders.

## Executive Summary

Wally is an interactive and fully autonomous item retrieving robot. He sits at his home location until a user prompts him to go and retrieve an item. The user tells Wally what they would like him to fetch from the “warehouse” and where they would like that item to be taken. Once he receives the request Wally goes into the warehouse picks up the specified item and takes it to the correct area. He then returns home and waits for his next request.

Once Wally receives a request via his on platform keypad, he sets off into a simulated warehouse. He navigates this warehouse using a line tracking algorithm, where he follows a black line on a white floor. As he traverses the warehouse he searches for the requested item using Radio Frequency Identification tags, or RFID tags. The rooms themselves are marked with the RFID tags, the requested item resides within a specific room.

Once Wally finds the room where the requested item is residing, he turns left and faces the room. Using his sonar rangers he centers the item into his field of vision and lowers his gripping arm. He then clasps the object and lifts it above his head. Wally then turns around, reacquires his line and continues through the warehouse until he finds the drop point, again marked by a RFID tag. Once he finds this point he does not turn left, but instead turns right and drops the item off. Using this system Wally picks up items on the left and drops them off on the right. Once Wally lets go of the item he turns around, reacquires the line, and continues through the warehouse until he returns to his home position. When he reaches his home position he stops, waits 10 seconds and then waits for another user to request an item in the warehouse.

As Wally is driving around the warehouse he is always conscience of unexpected obstacles that may block his path. He comes to a dead stop if an item crosses into his path and does not move. He then waits for this obstacle to be removed before proceeding forward with his business. He employs a Sonar Ranging sensor at the front of his platform to accomplish this task. Wally can also sense collisions with his bump sensors. If a collision is encountered then Wally immediately stops, backs up, and waits for any obstacles in his way to move. These behaviors help Wally accomplish his ultimate task of retrieving items.

Finally, at any given point in time Wally's mind can be read by looking at the LCD screen on the top of his platform. Everything that Wally is currently thinking will appear on this LCD screen. What he is waiting for, looking for, or already acquired is displayed here.

# Introduction

There is a growing interest among companies to process customer orders autonomously. Companies such as Wal-Mart, Publix, and Amazon, to name a few, are very interested in allowing customers to order their items from the internet instead of having the consumer visit their store. This would greatly reduce the cost of theft, and would allow them to be operated out of centralized warehouses rather than fancy store fronts. Using robotics these companies could work 24 hours and have minimal downtime, meaning that at anytime you place an order online it will be filled shortly rather than waiting until the next business day. This would increase profits substantially for the company. I feel that based on this approach my robot has an industrial purpose and someone may actually be interested in buying a robot like this.

My robot will also use a technology called RFID, in which the robot can identify an item just by being in proximity of that item. This would mean that the robot would not have to scan every item, but merely drive by items and read the RFID tags without stopping, until the specific tag is found. This is an upcoming technology that is still in its infancy, and consumer companies are very interested in this identification system. Wal-Mart and the Department of Defense are at the forefront with support for this technology. I am very interested in this form of identification. I think it quite possibly could replace keys, credit cards, ID badges, etc. This of course won't be for a while since the technology has yet to be perfected to such a secure level. I was very interested in this technology and I found a low cost reader with unique ID tags, so I decided to add it to my robot. This will allow me to delve into my interests and make a robot which serves a useful task.

Finally this paper is organized in a logical way that discusses all facets of my robots' design. I start by talking about the entire system and graphically show how the overall system is constructed. Then I move into a discussion on my mobile platform, its design, and method of construction. After that I talk about my robots actuation regarding how it moves around. Then I give a detailed sensor report followed by an account of all behaviors implemented. I finally close with my results, conclusion, recommendations, all documentation, and source code.

## Integrated System

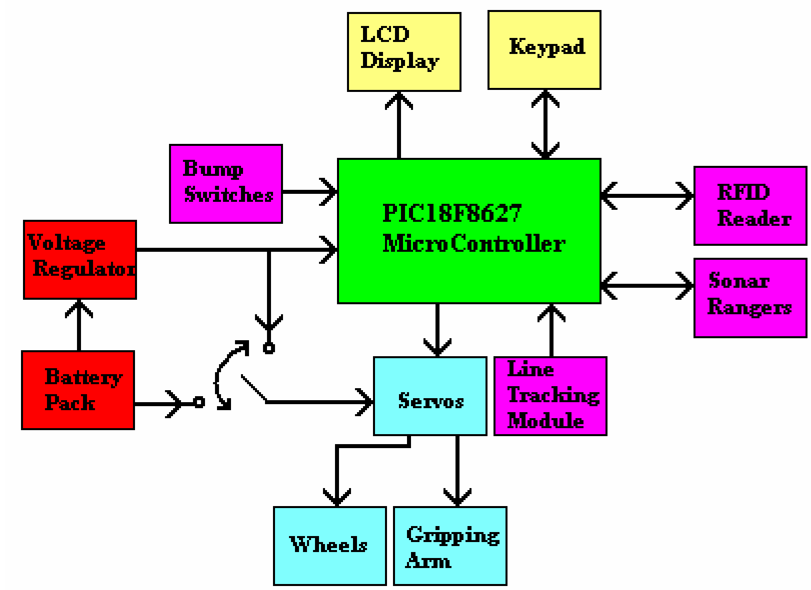


Figure 1: System level block diagram

Figure 1 shows a block diagram depicting the system level design of Wally. It shows in which direction information flows from the microcontroller and is color coded by function. Each function can be thought of as a subsystem to the overall project. The Green box represents the brain of the system and is the microcontroller; this is where the actual ‘thought’ process of the robot takes place. It processes signals from all of the sensors and operates all of the devices attached to it. The red boxes are the heart of the system and represent the power subsystem. Notice that this subsystem was designed to allow the servos to operate off of the same 5V regulated power as the rest of the robot. It can however be switched to draw power directly from the main battery or a secondary battery. This was implemented with the intentions of upgrading to a DC motor or being able to use this controller in another robot application which uses DC motors. The blue boxes are the actuation subsystem and perform all of the robots movements. The purple boxes are the sensory subsystem, which takes the human environment and transforms it into electrical signals. This enables Wally to decide for himself what course of action to take. The yellow boxes are the user interaction subsystem. This is how the robot receives instructions and gives feedback to the user.

Please note that the Line tracking module will be designed and built by me. I will use William Dubels' line tracking document as a reference. This module will consist of IR modules and voltage comparators.

## Mobile Platform

The platform that was designed for Wally was created with aircraft grade balsa wood and was designed in Protel 2005. It was developed with maneuverability and object lifting simplicity in mind. This dictates a small width, tall platform. The small width allows Wally to take corners easily and the tall design gives him the ability to lift normal size objects. With this in mind the platform consists of 5 tiers. Each tier is a rectangular shape that is 3" long by 2" wide. Notice that the width is smaller than the length, which sticks with the original design for maneuverability. Each tier is separated with a 1" metallic spacer which gives the platform a height of 5" not including the wheel height, which is 2". Most of the sensor mounting holes were not planned in Autocad, but were drilled out with the use of a Dremel tool once the platform was cut out. The LCD, On/Off switch, and tier to tier mounting holes were included in the Autocad files and were cut out to precise specifications to ensure that the tiers would be perfectly stackable.

After this platform was designed and the wood was cut I was stuck with the task of actually mounting my line trackers, wheels, servos, RFID reader, and casters to the bottom of this platform. Needless to say, this was not accomplished without modification to the bottom tier. I eventually had to increase the width of the bottom tier by moving the wheels out to the right and left by 1" each, thereby effectively increasing the width of the bottom tier by 2". This was accomplished with the help of "L" braces which then attached to "L" brackets that attached to the servos. Once the servos were successfully moved out of the way, two marble casters were used to stabilize Wally and allow him to turn in any direction. These casters were mounted in a straight line along the X-axis of Wally, basically one in the front center and one in the back center of the bottom tier. This allowed Wally to be totally stabile, while allowing movement in any direction. The RFID reader and line trackers were then easily mounted to the underside of the bottom tier.

In the end each of the tiers contains an important part of Wally, these tiers and what they include are given here. On the top tier there is a LCD, On/Off Rocker switch, and keypad. The next tier holds the gripping arm servos and the serial LCD piggy back board. The third tier is where the microcontroller is mounted. The fourth tier has the 9.6V battery pack and sonar rangers. Finally the fifth tier holds the servo battery pack and the bump switches on the top. Under the fifth tier is the line tracking board, RFID reader, servos, and two marble casters. Overall the intent of the design of the platform was observed in practice. Wally does indeed maneuver very well, and can lift items.

Hard lesson learned: simply making the platform small hinders the ability to attach peripherals to the underside of the robot even though it increases mobility. This then required many hours of playing with hardware and trying different configurations to simply get everything to fit. It would have been better to try and design the platform with system hardware location in mind. Not just build the platform and attempt to fit everything onto the given space. This simply did not work out very well and in the end I spent many hours modifying my platform with metal brackets that I purchased after walking around Lowe's for a while.

# Actuation

Wally employs a microcontroller development board, which I have designed, based on the PIC18F8627. This board has 5 slots dedicated to interfacing servos to the PWM channels on the microcontroller. Wally uses two standard servos, GWS S03N, purchased from Acroname to enable him to move around. These servos provide 3.40 [kg cm]/ 47 [oz in] of torque with a rotation speed of .23 secs / 60 degrees. They were hacked, by me, to allow continuous rotation. By hacking the servos it is meant that they will no longer have a feedback mechanism or mechanical stop in place. By removing these pieces of the servo, the servo will continue to turn never reaching its desired position and therefore enabling continuous rotation. This continuous rotation is required in order for Wally to actually move forward and backward. These servos are operating on a 6V battery supply that is separate from the 9.6V microcontroller board supply. At first one battery pack was attempted, but the servos would cause the entire board to reset after a certain amount of time running. Once the servos were put onto a separate supply they operated as intended and the board was no longer reset.

I originally attempted to use a DC motor, Tamiya 70168 Double Gearbox, and DC motor drivers to actuate Wally. I followed William Dubels' motor driver design even ordering the same H-bridge chips as him, but still I had no luck with DC motors. I used a DC power supply to ensure proper operation of the motors, but as soon as I hooked it up to a battery supply, the batteries were drained in a matter of about 20 seconds and I decided to return to servo actuation. I soon learned that the motors I chose to use were designed for high current low voltage operation, at most 6V operation, but the motor driver board I built needed at least 7.5V to operate properly. Making the DC motors incompatible with my motor driver board. This was probably the cause of my problems. I therefore use 2 servos to actuate Wally.

I used a turning in place algorithm to implement actuation and turning. Originally I had one wheel totally stop rotating while the other wheel rotated forward to implement my turns. I realized that this kind of turning increased my turning radius and reduced the maneuverability I had wanted to design into Wally's platform. Instead I chose to use an algorithm that allowed Wally to essentially stay in one place while turning. This "Turn in place" algorithm was accomplished by letting one wheel rotate forward and instead of turning the other wheel off it was set to rotate backward. This allowed the turning radius to become really small and Wally was then able to make turns about his axis of actuation. This was a very good and welcome discovery, it cleaned up my turns and made them look more smooth.

Finally, Wally uses two Joinmax digital servos from Pololu to enable actuation of a gripping arm. These servos supply more torque than the actuation servos, they are specifically made for robotic joints and lifting applications. The specifications of these servos are given as: 3.5 [kg cm] / 48.6 [oz in], with .18 secs/ 60 degrees. These servos are well suited to lift robotic size items.

# Sensors

The sensory capabilities of a robot will define what it can and can not do. The more sensors a robot has the more advanced it can become. Each of my sensors was chosen for a specific task. My development board was designed with 5 slots for sensors. Each slot contains a +5V and GND connection for the sensor. The signal can then be attached to any open pin on the microcontroller. A list of my sensors implemented is given below.

## 1) Ultrasonic (Sonar) Rangers

Wally employs several kinds of technologies to help him manipulate the human world. The first sensor of importance is the Devantech SRF05 Ultrasonic Ranger. This device basically works as his eyes and allows him to “see”. The ranger sends out a short burst of sound and awaits an echo back. The echo comes from devices in his field of vision. Based on how long it takes for the sound to come back the distance to the closest object could be determined. Wally uses one of these sensors, since he is a line tracking robot he will be concerned with objects directly in front of him, therefore one Ranger will be placed on the front in the center. Wally uses these sensors for two purposes. The first purpose is to identify obstacles in his path so that he can avoid them. The second purpose is to align an object he wishes to pick up directly in the front center of his gripping arm. The method of operation of this sensor as well as some experimental data is given below.

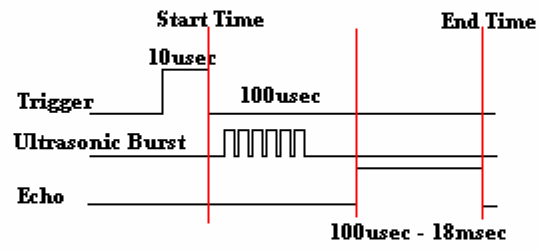


Figure 2: Sonar Ranger Operation

The operation of the Sonar ranger is shown in figure in 2. The operation basically works with two pins, the first pin is an input into the unit called the trigger pin and output from the unit called the echo pin. The microcontroller tells the unit that it wants to check for objects in front of Wally. This is done by making the trigger pin go logic high. Once this pin goes high for longer than 10μsec and then pulled back to logic low, the Ranger will send out an ultrasonic burst of sound at the falling edge. It is at this edge that all distance counting should begin. Once this sound burst is sent out, the unit will wait for 100μsec before making the echo pin go high. The unit then waits for the sound waves to bounce back to it. Once the sound bounces back to the unit the echo pin will be made a logic low value. At this falling edge is when the distance counting routine should be stopped. The distance is then proportional to the time delay between the falling edge of the trigger pin and the falling edge of the echo pin. If the echo comes back before the echo pin is ready or if no objects are in front of the robot then the unit will time out after 18msec. A table of my experimental data is given below.

| **Distance** | **Count Value** |
|---|---|
| 1" | 17 |

| | |
|---|---|
| 3" | 51 |
| 6" | 99 |
| 8" | 132 |
| 12" | 199 |
| 18" | 297 |
| 24" | 362 |
| 36" | 531 |
| 48" | 716 |
| 156" | 3407 |

Figure 3: Experimental Distance Measurement

The ranger was found to hold these internal values very well. At these distances I repeatedly obtained the same value. After 4 feet, 48”, the cone of the sonic burst became too large and would skew all of the values based on what was in the room. The time out amount was determined by placing an object right next to the unit so that the sound waves would be bounced back before the unit was ready, therefore effectively causing the unit to time out. This number was also stable at 3407, in figure 3 this value corresponds to the value of 4 meters, approximately 13feet or 156”. The value of 4 meters is given as the units maximum distance in its data sheet. A graph of this behavior is shown below, with the intermediate distances predicted by a linear line. The student determined that the closest reliable distance to accurately observe an object was about 3” and therefore bump switches should be used at a closer range than this to ensure collision detection in this range.

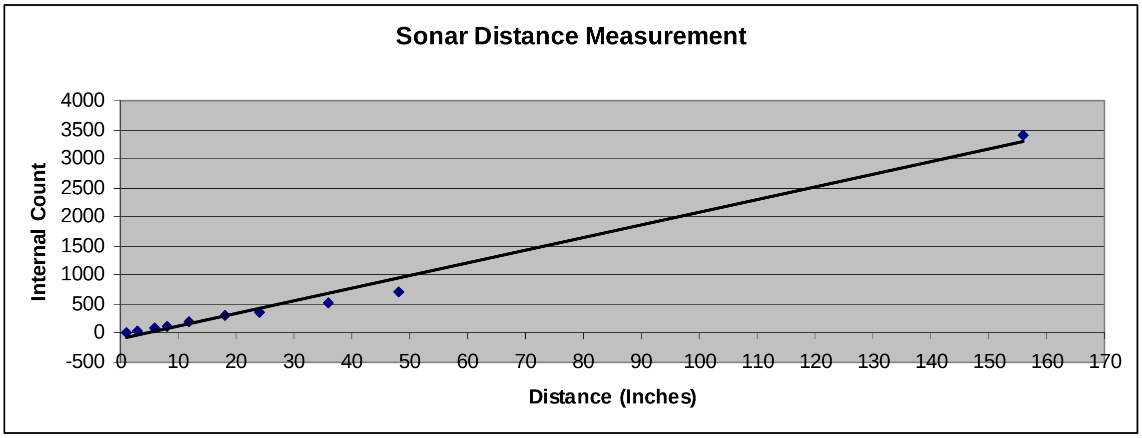


Figure 4: Graph of the Sonar Distance Measurements

## 2) Optek OPB745 IR Line Tracking Modules

Next Wally will be able to follow a line through a simulated warehouse. To allow Wally to navigate this line he needs a method of being able to "see" the line. This line tracking behavior is based on the Optek OPB745 IR emitter and detector module. This module has an infrared diode and a phototransistor in one package. The student determined that he will be using Wally on a white floor with a black line, these are two totally opposite color profiles and therefore can be easily differentiated using a comparator. A schematic of this circuit is shown below:

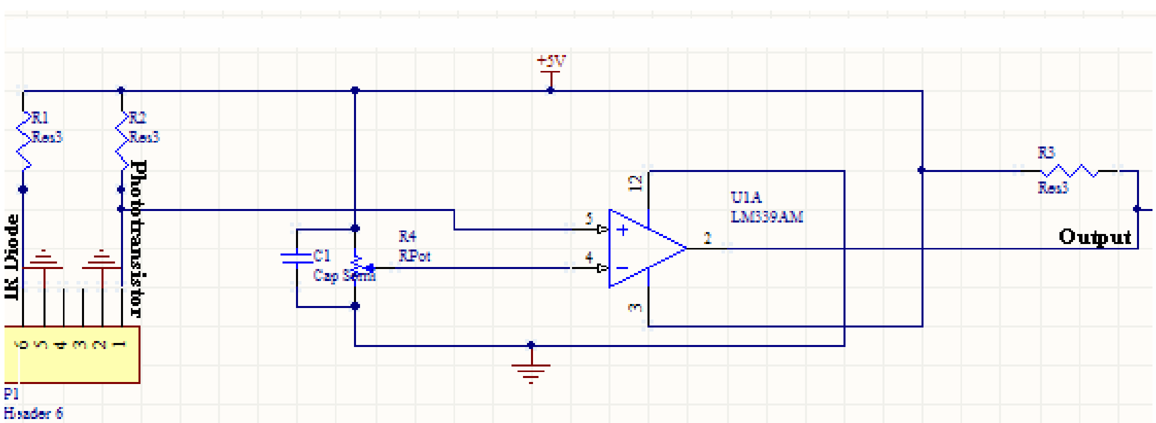

Figure 5: Line Tracking Circuit Schematic

In figure 5 the line tracking circuit is shown. This is a Protel schematic so the header at the bottom of the figure represents the Optek IR module. The anode of the IR diode is placed at Pin 6 while the cathode is placed at pin 5. The phototransistor has its collector on pin 1 and its emitter on pin 2 of the same header. This module is not available as a part in the schematic library of the program. This circuit works by measuring the reflected IR from the phototransistor, the more reflection it has the higher the voltage coming out of the transistor branch of the circuit. This voltage is then compared against a threshold, set by the potentiometer, and if it is above the threshold the device sees that there is a high reflection and so it must be on a white surface and a logic "0" is passed out of the comparator. If the voltage is below the threshold then the device sees a dark surface, since most of the IR is absorbed, and sets the output to a logic "1". In this manner Wally is able to track a dark line on a light surface. This circuit was adapted from William Dubel's line tracking documentation.

This circuit will allow the student to adjust the threshold at which Wally sees a dark surface and a light surface. This can be used to adjust Wally to slightly off white floors, but it should remain as close to white as possible for best results. The student has also found that this threshold helps adjust the robots IR distance. By changing this potentiometer value the robot could still identify a white or black surface at higher distances. Wally was able to accurately differentiate a white and black line from 3cm to about 1" by adjusting the potentiometer.

Since Wally is able to turn left and turn right, he employs 4 of these circuits; Two centered on the black line to allow him to find the line if he skews off of it, and then one on each side to identify intersections in the warehouse. This 4 IR configuration is the optimal way to have Wally negotiate turns and stay on the black line.

## 3) RFID reader and ID tags

Wally has the unique ability of being able to identify individual rooms by RFID technology. RFID stands for **R**adio **F**requency **ID**entification. He uses the RFID reader by Parallax to read unique passive ID tags. This device works on the principle of induction. The antenna on the tag reader induces a current into the antenna of the tag which powers up a microchip located inside it. This microchip then repeatedly transmits a 10bit unique ID number wirelessly, at a low frequency of 125 kHz. This is then read by the reader and output in a standard serial string to the microcontroller at 2400baud, 8 data bits, no parity bits, one stop bit, and one start bit. The unique ID string is 12 bytes long since it contains a start byte, stop byte, and 10 bytes of tag ID. The start byte is 0x0A and the stop byte is 0x0D which represents the line feed and carriage return values in ASCII respectively. The RFID reader initiates all of the communication, and does not accept any input or commands from the microcontroller. The students program waits to receive the start character and then stores the data until it reaches the unique stop character. Once this ID is read Wally then decides whether this room contains the object he needs or not. The student has successfully read ID tags at distances of 0 “to 3”. The datasheet for this device claims that it is capable of reading tags from 4”, but the student was unable to reliably attain this distance.

## 4) Bump Switches

Referring to the above discussion on the Sonar Rangers, it is easy to see that there may come a time when Wally can not see an object because it is either too close or out of the field of vision of him. Once this happens then a collision with the object in front of the robot is eminent and unavoidable. Since this is going to happen Wally needs a way to tell when he actually runs into or collides with an object. Wally has three bump switches, to implement collision detection. These are normally open (NO) lever action switches and when pressed the circuit is completed and the output is tied high. These are very simple devices to use and are very effective at what they do. Three switches are mounted on Wally, one in the front center, and one on each side of the front center. These switches are mounted low on Wally, below the field of view of the Sonar Rangers, on the bottom tier of the platform. The levers on these switches act as bumpers, like a car, so that Wally knows when he has collided with an object by checking the voltage at these pins. Once he collides with an object Wally backs up and waits until his path is clear before attempting to move forward again.

## 5) Keypad

Wally has the ability to interact with the outside world using a keypad that is attached to the top of his platform. This keypad allows users to tell Wally what item they wish to retrieve in the warehouse. That item can then be picked up and dropped off at any point in the warehouse and those locations are relayed to Wally through his keypad. This is a key sensor since it allows Wally to interact with the human world. Without the keypad the main goal of the project could not be achieved.

# Behaviors

## 1) Line Tracking

Wally is able to follow a line through a simulated warehouse. This line guides Wally to each room in the warehouse and allows him to choose a specific item. This ability is accomplished using the Optek OPB745 IR modules, listed in the sensor section above, with voltage comparators. Wally checks his line tracking sensors in a polling fashion based on the center two sensors. When he sees that both sensors are centered on the black line he moves forward. When the left sensor is off of the line and the right sensor is on the line he shifts right, and vice versa shifting left. When both sensors are off of the black line he turns to his left until his left sensor reacquires the line and then the shifting routine takes over, which readjusts Wally onto the black line. This allows Wally to make perfect 90 degree turns at the end of each line. When Wally wants to make a turn at an intersection he first looks at the center sensors to ensure that they are aligned. Once he is sure that they are aligned he knows that he found an intersection and then can make a decision on how to negotiate it.

## 2) Collision Avoidance

When Wally moves around his environment, or his “warehouse”, he will inevitably come across an obstacle in his path. When he encouters these obstacles he needs to first detect them and then react to them. Wally is constantly looking ahead of him using his Devantech Sonar Rangers discussed in the sensor section above. These sensors are implemented using an input capture methodology. Once the trigger pin is released the timer register is read and then the input capture bit is polled until the echo is returned back to the microcontroller. Once the echo is received back the captured previous timer register is subtracted from the captured register to give the total delay. This delay has a direct relation to the distance to the closest object. When an object is deemed too close, within 6” of him, Wally stops moving and waits for the obstacle to clear his path. Once the path is cleared Wally then continues on with his business. In this way collisions can be avoided and collision avoidance is in place.

### 3) Collision Detection

Wally can not rely on his sonar sensors to “see” everything much like humans can not rely on just vision alone. Wally needs a way to know when he is physically “touching” something in the world. He can achieve this sense by using bump sensors. These bump sensors tell Wally when he has hit an obstacle and should not continue moving forward. Wally has three bump sensors, all of which are oriented to the front of his mobile platform. The first sensor is in the dead center and can detect head on collisions. The other two sensors are on the sides, one on the left and one on the right. When he side swipes or clips an item he needs to be able to realize this, because these items outside of his peripheral vision can cause serious damage to his systems. These bump sensors are ideally in place to detect those obstructions which lie below the field of view of the sonar sensors or are too close to Wally for the sonar sensors to detect them. The pin that the bump sensors are connected to is pulled to logic high by a 470 ohm resistor. When the bump switches engage, this line gets tied to logic low causing a detectable voltage change on that pin. It is in this manner that collisions can be detected.

Once Wally detects a collision he stops moving forward, moves backward for about 1.5 seconds and then waits for the obstacles to clear his path before continuing forward. It is this behavior that shows collision detection.

## 4) RFID Scanning

Wally has the unique ability of being able to retrieve a specific item. In order for him to find a specific item he needs to be able to differentiate it from all other items available. Many systems in the past have used different types of technologies including bar code scanning, CCD cameras, and even infrared detection. W ally is unique since he uses a technology known as Radio Frequency Identification (RFID). This technology was explained in the sensor section above. Wally drives around his warehouse reading these RFID tags, which are embedded in the floor. These tags identify simulated rooms. Inside these rooms lie items that are being sought after by the user. Wally employs a polling mechanism to read each RFID tag. As he drives around he looks to see if he is receiving any type of communication on the RFID reader, if he is then he reads it and determines if that’s the item he wants or not. If its not then he continues on until he finds what he is looking for. In this manner RFID scanning is implemented.

## 5) Specific Item Retrieval

Wally has a keypad located at the very top of his platform. With this keypad a user can enter what item they want Wally to retrieve, and where they want him to place that item. Once Wally receives this request he will drive out into the warehouse using the line tracking, RFID scanning, collision avoidance, and collision detection behaviors described above to find that particular item. Wally drives around until he finds the RFID tag which contains the item that he is interested in collecting. Once he finds this tag he then stops his servos, turns to the left until he finds the object. The warehouse is set up such that all items to collect reside on the left, and the spaces to place items reside on the right. Once he finds this object using his sonar sensors he centers the object, opens his gripping claw and lowers his hand. Once his hand is lowered he closes his grip, but maintains the grip with a PWM signal. He then lifts his arm up and picks the item up. Once he has grasped the item he turns back toward the line and continues to track it, looking for the drop point. Once he finds the drop point he turns to his right and places the item down then opens his grip and raises his arm. At this point the item retrieval behavior is completed and Wally returns to his home position to wait for the next item to retrieve. This behavior makes use of just about all of his sensors as well as PWM signaling. This behavior above all others took the longest to implement and debug.

## Experimental Layout and Results

The final version of Wally relied on implementing calibration programs to understand the full ranges and values that can be obtained from each sensor. The servos needed to be calibrated to find forward movement and backward movement, plus how to change the speed of each wheel to get them to match, and closely match. Once the test programs had been written per sensor it was time to integrate several behaviors at once. I started by combining my actuation algorithms with my line tracking algorithm. This could have gone more smoothly, but I ran into problems with cables that had been made as well as with traces on the board that I had milled. These were constant battles that I had fought until I finally made a new board with thicker, sturdier traces. Then I found that a cable that I made had a bad connection in the middle point. Once this was overcome line tracking and actuation went smoothly. That was until my board decided to keep resetting after it made a 90 degree turn. It was at this point I changed to a two battery pack power system. This solved that problem. Then I added my collision avoidance behavior and gripping algorithms. At this point I started trying to have my robot pick items up. This took A LOT of time and effort to center the object and actually pick it up. Then of course I implemented the user interface and specific item retrieval behaviors. I ran a lot of little programs and “experiments” to find values for my actuation servos, gripping servos, sonar rangers, LCD settings, keypad captures, and line tracking algorithms. It was these smaller programs that helped me to understand how the sensors worked and ultimately build my final product. I would say that these experiments were a success since I have a fully functioning unit.

## Conclusion

Overall I learned an enormous amount from this class. I came into this class knowing a lot about electronics and circuit design, but I feel it was my confidence in this area that ultimately led to my failure. It was not a total failure since I completed the robot by the end, but final demo could have gone a lot better. If I would have accepted a simpler approach and went with polling algorithms since the beginning then I would not have had the trouble I did. I attempted to do an interrupt based system, and interrupts in a basic language are not user friendly. I found that there were flawed instructions in the basic language, ones that don’t work well with others, and are available solely on their own. It took a while to realize that, but once I did and tried not to be smarter than the program I was able to accomplish a lot more. I rewrote my code to maximize the polling algorithm and minimize wasteful loops, this eventually allowed me to accomplish my final product. This class definitely tests your design and debug skills.

Now as far as my robot is concerned realistically I accomplished a basic robot that does line tracking, collision avoidance, RFID scanning, and item retrieval. I would have liked to make the robot wireless and removed the keypad, but getting the rest of the robot working was enough of a problem to overcome. I feel that my biggest weakness was the lack of planning when I designed the body. If I would have laid everything out in CAD then I wouldn't have had to add the extra hardware to move my wheels out to the sides and try to squeeze my RFID board underneath the robot. I didn't understand at first the time commitment and frustration involved in building a basic robot, but I do now. It's been a very time consuming and at times frustrating class, but overall it has given me one of the best educational experiences I have ever encountered. The skills I have learned in this class are practical hands on skills that no other theory based class can give you. I had to rely on my design skills, creativity, debug experiences, and at times raw luck to achieve all of the goals in this class.

I feel that on demo day I wasn't prepared to show my robot and that was further accented by the fact that things which worked earlier on, stopped working. This was a frustrating time, but it gave me what I needed to forget about trying to achieve everything based on interrupts and get back to polling routines since robots are interacting with the human world, they really don't have to react on the nanosecond. It did involve some creative design because the robot can not be stuck in wait loops and still expect to react to RFID tags and line tracking routines at the same time.

Overall if I had to do the robot over I would have gone with servos from the beginning and not mess with DC motors, however DC motors as well as wireless connectivity would be an upgrade. I would also send my line tracking board out to be professionally fabricated, this would have alleviated A LOT of problems early on. I would have designed my platform better and accounted for at least all sensors on the bottom of the robot as well as the servos and wheels. I never realized how long it took to make cables until I had to make a lot of them. I feel that I started early and I was on top of my game since I made my own boards and acquired all of my parts early on. It was my job and other classes that stuck me behind.

In conclusion I have learned a lot about myself and about electronics through this class. It has improved my debugging ability and my electronics design skills. I highly recommend this class however it is not for the faint of heart because it requires a lot of time and dedication. The biggest thing I learned in class is that EVERYTHING takes 300x longer than expected so plan for it.

# Documentation

**Microchip PIC 18F8627 Datasheet**
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2057&ty=&dty=Data+Sheets&section=Data+Sheets&ssUserText=PIC18F8627

**Machine Intelligence Laboratory University of Florida**
**www.mil.ufl.edu/imdl**

**William Dubel Line Tracking**
www.mil.ufl.edu/imdl/handouts/lt.doc

**William Dubel Motor Drivers**
http://www.dubel.org/motordriver/

Parallax RFID Reader and RFID Tags Documentation
http://www.parallax.com/detail.asp?product_id=28140

**Acroname SRF05 Sonar Rangers**
http://www.acroname.com/robotics/parts/R271-SRF05.html

**Pololu Digital Gripper**
http://www.pololu.com/products/joinmax/0093/

**Sparkfun LCD Serial Piggy Back**
http://www.sparkfun.com/commerce/product_info.php?products_id=258

# Appendices

**Final copy of code**


```
'*******************Globals***********************************
        STATE  VAR BYTE             'State of the mainb loop for WALLY
'***********LCD***************************
        LCD    VAR PORTG.2          'Mask the port where the LCD is
located
'***********Line Tracking*****************
        mstate VAR BYTE             'Variable used to control the state of
the motors
        RC     VAR PORTD.2          'Right Center Line Sensor
        LC     VAR PORTD.1          'Left Center Line Sensor
        RL     VAR PORTD.3          'Right Edge Line Sensor
        LL     VAR PORTD.0          'Left Edge Line Sensor
        TL     VAR BIT
        TR     VAR BIT
        SHL    VAR BIT
        SHR    VAR BIT
        PERP   VAR BIT
'***********RFID Scanning*****************
        RFEN   VAR PORTC.1          'RFID Enable PIN
        RFID   VAR PORTC.7          'RFID Data Pin
        UID    VAR BYTE[10]
        HDR    VAR BYTE
```

```
        TAIL   VAR BYTE
        TAGnum VAR BYTE
        source VAR BYTE
        dest   VAR BYTE
        found  VAR BYTE
        TAG1  CON "F"
        TAG2  CON "E"
        TAG3  CON "9"
        TAG4  CON "5"
        TAG5  CON "B"
'***********Gripping Arm.******************
        duration VAR BYTE
        CNTR   VAR BYTE
        drop     VAR BIT
'***********Sonar Rangers******************
        THR          VAR WORD          'Collision Avoidance Threshold
        TRIG   VAR   PORTE.1           'Trigger Pin
        ECHO VAR   PORTC.2             'Echo Pin
        DIST   VAR WORD          'Distance to object
        DIST1 VAR   BYTE
        SNDPNG       VAR BIT
        DELAY   VAR BYTE
'***********Bump Switches******************
        bump   VAR   PORTB.0
'***********Keypad************************
        PAD    VAR BYTE[17]
        ROW          VAR BYTE
        COL          VAR BYTE
        pushed  VAR BYTE
        get1    VAR BYTE
        nokey  VAR   BIT
        key          VAR BYTE
        PAD[0] = "D"
```

```
        PAD[1] = "C"
        PAD[2] = "B"
        PAD[3] = "A"
        PAD[4] = "#"
        PAD[5] = "9"
        PAD[6] = "6"
        PAD[7] = "3"
        PAD[8] = "0"
        PAD[9] = "8"
        PAD[10] = "5"
        PAD[11] = "2"
        PAD[12] = "*"
        PAD[13] = "7"
        PAD[14] = "4"
        PAD[15] = "1"
'**********************End of Globals***************************
'**********************Pic Initialization**********************
            OSCCON  = $63       'Intialize oscillator to 4MHz
            ADCON1  = $0F       'Make all pins digital I/O
            TRISA   = $00
            TRISB   = $01
            TRISC   = $84       'RC2 is Sonar Echo Input, RC7 is RFID
Input
            TRISD= $FF  'Set the inputs for the line tracker
            TRISE   = $00 ' The Sonar Rangers, E1 is Trigger Output
            TRISF   = $0F 'Keypad
            TRISG   = $00
            TRISH= $00
            TRISJ  = $01

            'Configure TMR1 to Input Capture and TMR2 to PWM Modes
            T3CON = $00
            T1CON = $C9 'Turn TMR1 on
```

```
T2CON = $27

'Initialize PWM channels for Servo
'CCP5CON = $0F
'CCP4CON = $0F
CCP2CON = $00
PR2     = $FF

'Initialize Input Capture For Sonar Rangers
CCP1CON = $04                   'ECCP1 is on RC2
PIR1.2 = 0
'Initialze the UART for RFID
BAUDCON = $00
TXSTA1  = $00
RCSTA1  = $90
SPBRG   = 25
PIR1.5  = 0                'UART Flag
'Enable Global and Peripheral Interrupts
'INTCON = $C0
Pause 3000
SerOut2 LCD, 84, [ $7C, 4,  $7C,  6, $7C, $1, $7C, $0D] 'Initialize
the LCD
SerOut2 LCD, 84, [ $FE,  $01, $FE, $02, $FE, $0D]

'*************************************************************

'********************Variable Initialization*******************
restart: STATE  = 0
        mstate = 0
        TL     = 0
        TR     = 0
        SHL    = 0
        SHR    = 0
```

```
                RFEN   = 0
                TRIG   = 0
                DIST   = 50000
                DIST1  = 00
                THR       = 1600
                SNDPNG = 1
                DELAY  = 0
                found  = 0
                PERP   = 0
                CNTR   = 0
                drop = 0
'*************************************************************
'***************************MAIN******************************
            GoSub gettags
Main:
            Select Case STATE
                Case 0 'Normal Operation and Line Tracking
                    'GoSub CLR_LCD
                    'SerOut2 LCD, 84, [$FE, 128, "Normal"]
                    IF bump = 0 Then
                        state = 3
                    Else
                        IF SNDPNG = 1 Then
                            GoSub PING
                        EndIF

                        GoSub Scan
                        IF PIR1.2 = 1 Then GoSub calcdist
                        IF DIST < THR Then
                            STATE = 1
                        EndIF
                    'SerOut2 LCD, 84, [$FE, 192, "STATE:", DEC STATE]
                        GoSub Track
```

```
        EndIF

    Case 1 'Stop Motors due to object detection
        mstate = "S"
        GoSub motors

        GoSub CLR_LCD
        SerOut2 LCD, 84, [$FE, 128, "Object Detected"]

        While DIST < THR
            IF PIR1.2 = 1 Then GoSub calcdist

            IF SNDPNG = 1 Then
                GoSub PING
            EndIF
        'SerOut2 LCD, 84, [$FE, 192, "STATE:", DEC STATE]
        Wend
        STATE = $00
    Case 2



        GoSub Track
        'IF PERP = 1 Then
            'mstate ="S"
            'GoSub Motors
        'EndIF

        IF SNDPNG = 1 Then
            GoSub PING
        EndIF

        IF PIR1.2 = 1 Then GoSub calcdist
        'THR = 1500
```

```
                IF DIST < 1400 Then
                    GoSub Track
                    GoSub Track
                    mstate = "S"
                    GoSub Motors
                    GoSub Opencl
                    GoSub Closecl
                    STATE = 0
                EndIF

            Case 3
                mstate = "S"
                GoSub Motors
                GoSub CLR_LCD
                SerOut2 LCD, 84, [$FE, 128, "Collision Detected"]
                While bump = 0
                Wend

            Case Else
                mstate = "S"
                GoSub motors
                GoSub CLR_LCD
                SerOut2 LCD, 84, [$FE, 128, "ERROR!!!"]
        End Select

        GoTo Main
'***************************************************************
'***************************Clear The LCD*********************
CLR_LCD: SerOut2 LCD, 84, [ $FE,  $01, $FE, $02, $FE, $0D]
         Return
'***************************************************************
'***************************Motor Arbitrator******************
Motors:
```

```
Select Case mstate
    Case "F"        'Move Forward
         CCP4CON = $0F
         CCP5CON = $0F
        'Servo1 ' Right Wheel
        CCPR5L = $12
        CCP5CON.5 = 1
        CCP5CON.4 = 1

        'servo2 Left Wheel
        CCPR4L = $96
        CCP4CON.5 = 0
        CCP4CON.4 = 0

        T2CON   = $27

    Case "B"        'Move Backward
        'servo1
        CCPR5L = $7D
        CCP5CON.5 = 0
        CCP5CON.4 = 0

        'servo2
        CCPR4L = $A
        CCP4CON.5 = 0
        CCP4CON.4 = 0

        T2CON   = $27

    Case "L"        'Turn/Shift Left
         CCP4CON = $0F
         CCP5CON = $0F
        'servo1
```

```
            CCPR5L = $12
            CCP5CON.5 = 1
            CCP5CON.4 = 1

            'servo2 off
            CCPR4L = $A
            CCP4CON.5 = 0
            CCP4CON.4 = 0

            IF(TL = 1) Then
                T2CON   = $27
                'Pause 500
                While (LC = 0)
                Wend
            EndIF

  IF SHL = 1 Then       'readjust until the right sensor finds the black line
                T2CON   = $27
                While(RC = 0)
                Wend
 EndIF

            TL = 0
            SHL = 0

           'T2CON   = $00
            CCP4CON = $00
            CCP5CON = $00

      Case "R"      'Turn/Shift Right
            CCP4CON = $0F
            CCP5CON = $0F
```

```
                    'servo1 OFF
                    CCPR5L = $7D
                    CCP5CON.5 = 0
                    CCP5CON.4 = 0
                    'servo2
                    CCPR4L = $96
                    CCP4CON.5 = 0
                    CCP4CON.4 = 0

                    'T2CON   = $27
                    IF(TR = 1) Then
                        T2CON   = $27
                            While(RC = 0)
                            Wend
                    EndIF
            IF SHR = 1 Then    'readjust until the left sensor reacquires the black line
                        T2CON   = $27
                        While(LC = 0)
                        Wend
                    EndIF
                    T2CON   = $00
                    TR = 0
                    SHR = 0
                Case "G"        'Get Item
                    CCP4CON = $0F
                    CCP5CON = $0F
                    'servo1
                    CCPR5L = $12
                    CCP5CON.5 = 1
                    CCP5CON.4 = 1
                    'servo2 off
                    CCPR4L = $0A
                    CCP4CON.5 = 0
```

```
        CCP4CON.4 = 0
        While DIST > THR
            IF SNDPNG = 1 Then
                GoSub PING
            EndIF
            IF PIR1.2 = 1 Then GoSub calcdist
        Wend
        Pause 150
        'Pause 1000
        CCP5CON = $00
        CCP4CON = $00
        GoSub Opencl
        GoSub Closecl
        'GoSub lift
        SHR = 1
        mstate = "R"
        GoSub motors
        'T2CON   = $00
         CCP4CON = $00
         CCP5CON = $00
    Case "D"        'Drop Off Item
         CCP4CON = $0F
         CCP5CON = $0F
        'servo1
        CCPR5L = $7D
        CCP5CON.5 = 0
        CCP5CON.4 = 10
        'servo2 off
        CCPR4L = $96
        CCP4CON.5 = 0
        CCP4CON.4 = 0
        Pause 1000
        CCP5CON = $00
```

```
                CCP4CON = $00
                GoSub Opencl
                GoSub lift
                SHL = 1
                mstate = "L"
                GoSub motors
                'T2CON   = $00
                 CCP4CON = $00
                 CCP5CON = $00
            Case "S"        'Stop the motors
                 'T2CON   = $00
                 CCP4CON = $00
                 CCP5CON = $00
        End Select

 Return
'*****************************************************************
'*****************Line Tracking********************************
Track:
    Select Case LC
        Case 0
            Select Case RC
                Case 0
                    TL = 1          'Turn 90 degrees to the left
                    mstate = "L"
                    GoSub motors
                    PERP = 0
                Case 1
                    SHR = 1 'SHift to the right
                    mstate = "R"
                    GoSub motors
                    PERP = 0
            End Select
```

```
        Case 1
            Select Case RC
                Case 0
                    SHL = 1         'Shift to the left
                    mstate = "L"
                    GoSub motors
                    PERP = 0
                Case 1
                    mstate = "F"    'Move forward
                    GoSub motors
                    IF LL = 1 Then PERP = 1
            End Select
        End Select
    Return
'****************************************************************
'******************Collision Avoidance**************************
PING:
    SNDPNG = 0
    TRIG = 1
    TMR1H  = 0 'Clear Timer1
    TMR1L  = 0 'Clear Timer1
    While (Delay < $20)             'Create an interruptable delay for 10us
        Delay = Delay + 1
    Wend
    PIR1.2 = 0  'Clear the interrupt flag
    TRIG   = 0
    DIST1  = TMR1L
    Delay = 0
    Return
'Set Local and Global Variable
'****************************************************************
'******************RFID Scanning********************************
Scan:
```

```
            SerIn2  PORTC.7, 396, 3, Leave, [wait($0A), STR UID\10, TAIL]
            GoSub Report
            'SerOut2 PORTG.2, 84, [$FE, 192," Found: ", UID[9]]
            IF UID[9] = source AND found = 0 AND drop = 0 Then
                mstate = "S"
                GoSub Motors
                found = 1
                GoSub grab
                mstate = "G"
                GoSub Motors
                GoSub CLR_LCD
                SerOut2 PORTG.2, 84, [$FE, 192,"PICKED UP"]
            EndIF
            IF UID[9] = dest AND found = 1 AND drop = 0 Then
                mstate = "S"
                GoSub Motors
                GoSub CLR_LCD
                mstate = "D"
                GoSub motors
                SerOut2 PORTG.2, 84, [$FE, 192,"Dropped OFF "]
                drop = 1
            EndIF

            IF drop = 1 AND UID[9] = TAG5 Then
                mstate = "S"
                GoSub Motors
                GoSub CLR_LCD
                SerOut2 PORTG.2, 84, [$FE, 128,"AT HOME!!!"]
                SerOut2 PORTG.2, 84, [$FE, 192,"DONE!!! "]
                CCP3CON     = $00
                duration = 0
donez:      PulsOut PORTG.0, 145
```

```
                    Pause 18
                    duration = duration + 1
                    IF duration < 51 Then GoTo donez

                    Pause 10000
                    GoTo restart

                    'While(1)
                    'Wend
            EndIF
Leave:
            Return
'**************************************************************
'******************Gripping ARM**********************************
Opencl:
                            duration = 0
DN:                         PulsOut PORTE.7, 145
                            Pause 18
                            duration = duration + 1
                            IF duration < 51 Then GoTo DN

grab:               duration = 0
                            CCP3CON     = $00
                            Pause 100
OP:                   PulsOut PORTG.0, 50
                            Pause 18
                            duration = duration + 1
                            IF duration < 51 Then GoTo OP
                            Return
Closecl:
                            duration = 0
                             CCP3CON = $0F
                             CCPR3L = $65
```

```
                CCP3CON.5 = 0
                CCP3CON.4 = 1
                Pause 2000
lift:       duration = 0
UP:             PulsOut PORTE.7, 50
                Pause 18
                duration = duration + 1
                IF duration < 51 Then GoTo UP
                Return
'*************************************************************
'*****************Sonar****************************************
calcdist:
            DIST = (CCPR1L - DIST1)  + $FF * CCPR1H
            'SerOut2 LCD, 84, [$FE, 128, "ENTRADA: ", DEC DIST]
            SNDPNG = 1
            PIR1.2 = 0  'Clear the interrupt flag
        Return
'*************************************************************
'****************************MAIN MENU*****************************
gettags:
                GoSub CLR_LCD
                SerOut2 PORTG.2, 84, [$FE, 128, "Pick Up At Tag"]
                SerOut2 PORTG.2, 84, [$FE, 192, "1..2..3..4"]
inval:      GoSub keypad
                Select Case key
                    Case "1", "2", "3", "4"
                        GoSub DIR_PU
                        GoTo getdest
                    Case Else
                SerOut2 PORTG.2, 84, [$FE, 128, "INVALID DIGIT"]
                        GoTo inval
                End Select
getdest:    SerOut2 PORTG.2, 84, [$FE, 128, "Drop Off At Tag"]
```

```
                    SerOut2 PORTG.2, 84, [$FE, 192, "1..2..3..4"]

inval1:         GoSub keypad
                    Select Case key
                        Case "1", "2", "3", "4"
                            GoSub DIR_DO
                            GoTo bye
                        Case Else
                    SerOut2 PORTG.2, 84, [$FE, 128, "INVALID DIGIT"]
                            GoTo inval1
                    End Select
Bye:        GoSub Report
                Return
'********************************KEYPADSCAN************************
'Scan keypad until a button is pushed
keypad:         get1 = $10
getkey:     For COL =0 TO 3
                    PORTF = get1
                    ROW = PORTF << 4
                    IF ROW != $0 Then
                        GoTo gotkey
                    EndIF
                    get1 = get1 * 2
                Next COL
                nokey = 1
                GoTo keypad
gotkey:     nokey = 0
                pushed = (COL* 4) + (ROW >> 5)
                IF ROW = $80 Then
                    pushed = (COL*4) + $03
                EndIF
Debounce:   While(ROW != 0)
                    ROW = PORTF << 4
```

```
                Wend
                key = pad[pushed]
goback:         Return
'************************************************************************
DIR_PU:
            Select Case key
                Case "1"
                    source = TAG1
                Case "2"
                    source = TAG2
                Case "3"
                    source = TAG3
                Case "4"
                    source = TAG4
                Case Else
                    source = $00
            End Select
                Return
DIR_DO:
            Select Case key
                Case "1"
                    dest = TAG1
                Case "2"
                    dest = TAG2
                Case "3"
                    dest = TAG3
                Case "4"
                    dest = TAG4
                Case Else
                    dest = $00
            End Select
                Return
Report:
```

```
GoSub CLR_LCD
IF found = 0 Then
    SerOut2 PORTG.2, 84, [$FE, 128, "Searchin, LF: "]
    Select Case source
        Case TAG1
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG1"]
        Case TAG2
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG2"]
        Case TAG3
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG3"]
        Case TAG4
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG4"]
        Case TAG5
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG5"]
    End Select
Else
    IF found = 1 Then
        SerOut2 PORTG.2, 84, [$FE, 128, "Picked UP!, LF: "]
        Select Case dest
        Case TAG1
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG1"]
        Case TAG2
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG2"]
        Case TAG3
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG3"]
        Case TAG4
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG4"]
        Case TAG5
            SerOut2 PORTG.2, 84, [$FE, 192, "TAG5"]
        End Select
    EndIF
EndIF
Return
```

End