Jay Cushing

IMDL Summer 2004
Written Report 2

Table of Contents

Abstract

This report documents the development of Atlas, my robot design for IMDL Summer 2004. Atlas' purpose and functions will be introduced, and then I will detail how I implemented these behaviors. Sensors, actuation systems, platform design, behavior and software design will be discussed. Several challenges arose during the construction of my robot and I will document these challenges and how I overcame them.

Executive Summary

Atlas is an autonomous mobile robot built for the Summer 2004 IMDL class at the University of Florida. Its purpose is to travel around a room and create a map that can be plotted on a computer and exported to popular drafting programs such as AutoCAD. Atlas is powered by the MAVRIC board which houses an Atmel ATMega128 chip. The chip features 8 A/D ports, I^2C bus, 4 timers, 2 USARTs and plenty of memory.

Atlas uses a Laipac 433 MHz transmitter/receiver pair to transmit data from the robot to the computer that collects the data. The computer runs data logging and plotting software that I coded using C. Data is collected and exported to an AutoCAD script which can be used to automatically create a draft.

Wall following and obstacle avoidance are implemented with Sharp IR distance sensors. Heading information is provided by a Devantech electronic compass module.

Mobility is provided by two servos modified for continuous rotation. They are controlled by generating PWM signals using one of the ATMega128's timer channels.

Atlas is powered by 2 NiMH battery packs. One, a 1000 mAh battery pack, is used to power the board and sensors. The other, a 1600 mAh pack, is used to power the servos. The RF receiver module is powered by a standard 9-volt battery.

Introduction

Atlas is a mapping robot. It acquires a wall of a room and follows it. As Atlas follows the wall, information about its position and heading are transmitted via RF link to a PC where the data is processed and a map is made. The software program exports the finished map to AutoCAD, which can make Atlas a convenient tool to use to survey a room and automatically create a draft in AutoCAD.

Atlas has several sensors comprising its sensor suite that assist with navigation. These sensors will be discussed in the following sections and include IR distance sensors, an electronic compass and the radio transmitter. These sensors help Atlas carry out its behaviors, which will be discussed in detail in the next section.

Integrated System

Figure 1 shows a block diagram of how some of the sensors and subsystems for Atlas are connected to each other. Atlas utilizes the A/D converter ports to read the voltages off of the IR distance sensors and convert it to a 10-bit digital representation. The timer is used to generate PWM signals that control the servos. Another timer channel is used to generate periodic interrupts to signal when to transmit information to the computer.

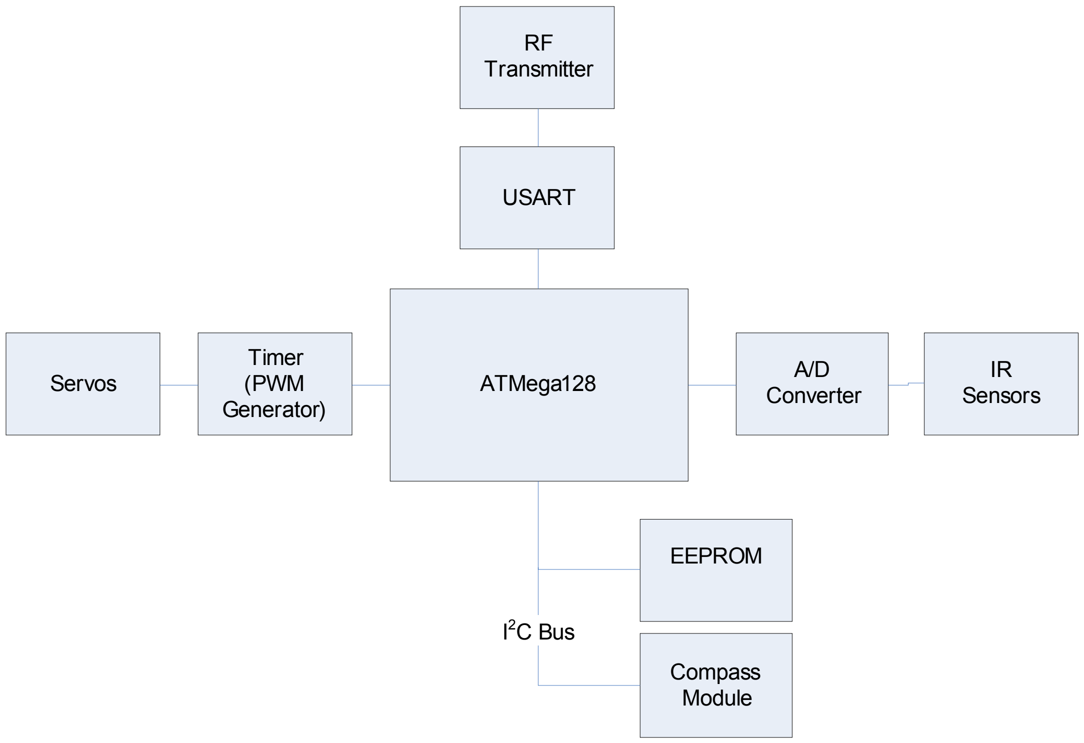

Figure 1: Atlas Simplified Block Diagram

The I^2C bus is used to read data from the digital compass module. The I^2C bus was very easy to use after learning the basics. One of the ATMega's USARTs was used to implement the wireless link used for telemetry.

Upon reset or power on, Atlas enters an initialization period. During this time, the devices are initialized. The value of the forward facing IR sensor is read. This value becomes the threshold for obstacle avoidance and wall following. This is the distance Atlas stays from walls and objects. The RF transmitter is initialized and data is sent to the computer about the current state. After 4 seconds, Atlas enters the main behavior pattern, which is wall following. Figure 2 shows a basic flowchart of how Atlas works.

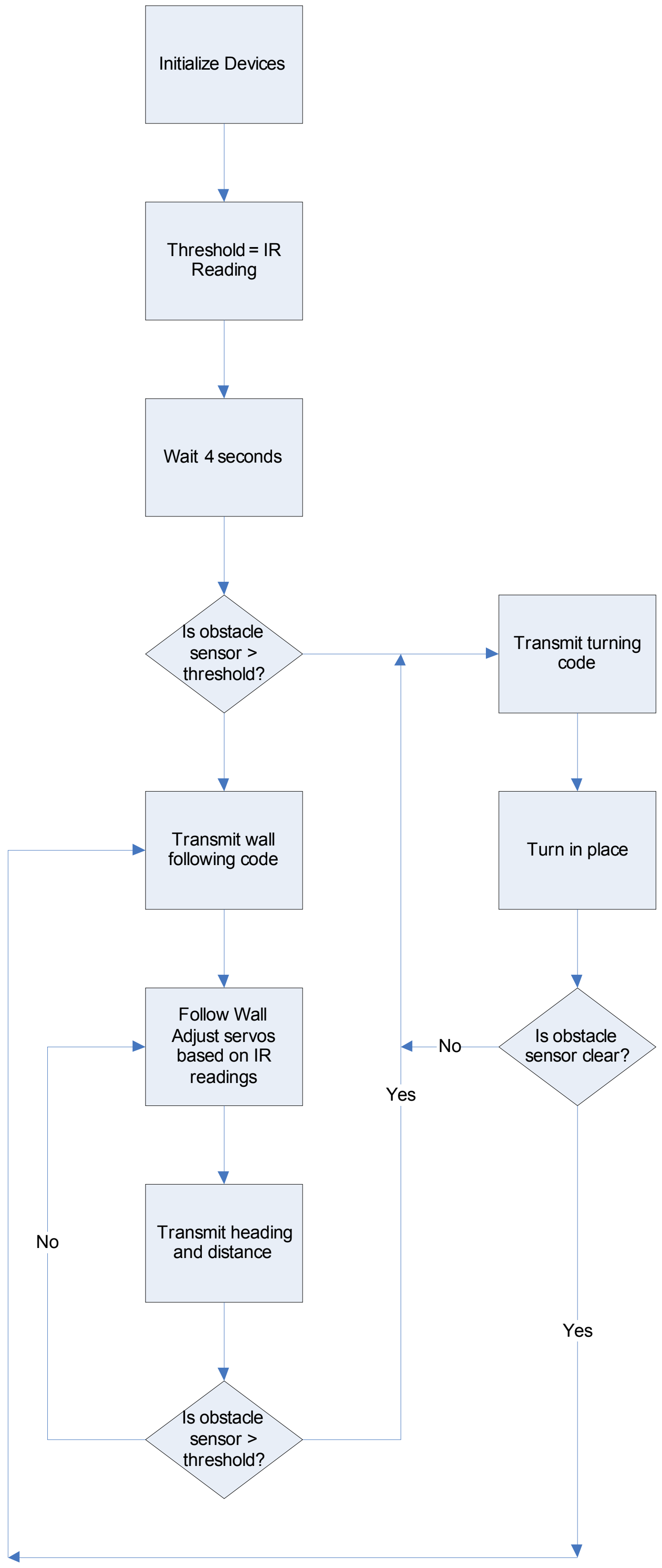
Initialize Devices
Threshold = IR Reading
Wait 4 seconds
Is obstacle sensor > threshold?
Transmit turning code
Transmit wall following code
Turn in place
Follow Wall Adjust servos based on IR readings
Is obstacle sensor clear?
No
Yes
Transmit heading and distance
No
Yes
Is obstacle sensor > threshold?

Figure 2

In my robot code, I have a global variable named ROBOT_STATUS which is set to defined values to indicate what mode the robot is currently operating in. There are five different defined states: Initialization (INIT), Wall Following (FOLLOW), Turning (TURN), Finding Wall (FINDING), and a Paused state which was used as an error state. The main body of code makes decisions based on the value of ROBOT_STATUS. The code follows the flowchart in Figure 2 closely.

Data is transmitted every 200 ms based on the timer interrupt. This is used as a rough odometer. By media day, the timer may be replaced by a wheel encoder, giving more accurate odometry. Every time the interrupt is fired, the handler uses the ROBOT_STATUS variable to determine what to transmit. All data uses a form of Manchester encoding to reduce errors. There are several codes that indicate what mode the robot is in to the remote computer. These codes are sent during the initialization and turning behaviors. A code sequence is also sent out when Atlas starts wall following.

If the current behavior is wall following, Atlas transmits heading and position (number of interrupts since last turn). This information is formatted into a data packet that will be explained in the Sensors section.

The PC code receives and decodes the incoming data. It keeps track of what state the robot is in. Every time a character is received and decoded, a large switch statement decides what to do based on the value of ROBOT_STATUS. Heading values are kept in an array until a turn is detected. Then the average heading and the total distance are computed and stored in a struct as follows:

```
struct map{
            int clicks;
            double heading;
      };

struct map path[255];
```

When computing the average, I first found that I was getting some errors from values wrapping around from 359 degrees to 0 degrees. I decided to transform the headings using sin(heading/2) and taking the average of those values. Then after averaging, I take the arcsin of the average and multiply by 2 to get the average heading.

When the user presses a key to finish mapping, the program compiles the results and makes an AutoCAD script that automatically creates an AutoCAD draft of the map. By Media Day I should have it plotting in real time.

Mobile Platform

For my mobile platform, I used a simple design. Atlas does not face any mechanical challenges in its planned purpose, so I chose to use a 2-wheel differential drive system. The platform is just a circle with the wheels on a center axis. I used small nylon coasters on the opposite axis to keep the robot level.

The MAVRIC board is mounted between the servos. I built a separate circuit board to mount headers and switches on. This board was mounted above the MAVRIC. A bank of IR sensors was placed above the right servo motor for wall following. The front of the robot has another IR sensor for obstacle avoidance.

I think keeping the platform design simple will eliminate any unnecessary hassles that more advanced designs might bring. Atlas does not need anything fancy in his platform design.

Actuation

The only form of actuation on my robot are the servos. I have 2 Futaba servos that give about 47 oz.in. of torque. They are powered with 5 volts off of the 1600 mAh RC battery. I modified the servos for continuous rotation.. They take a PWM pulse with a period of about 20 ms as an input. I found that the neutral for both servos was when the pulse was kept high for 1.5 ms.

The servos I used do not go very fast. I have considered upping the voltage but did not want to risk damaging them. Speed is not essential to Atlas' operation. In fact, slower speeds means less wheel slippage which makes for more accurate odometry.

To control the servos, I have a function set_servo(int servo, int speed) where servo is the number of the servo to set the speed for and speed is a number between -100 and 100. 100 is full speed forward and -100 is full speed reverse. The function computes the necessary PWM settings and outputs it.

In the while(1) portion of my robot code, I start every loop by setting variables for the motor speed equal to 80,80 which would have both motors going straight at a decent clip. Next decisions are made in increasing priority over what the motor speeds should be. The highest priority function is at the end of the loop, so it has the last say on what the motor speeds should be. The last statement in the while loop is the set_servo() function which sets the servos to the desired speeds.

Sensors

Atlas' sensor suite is designed for navigation. The first sensors are Sharp GP2D120 IR Distance Measuring Sensors. They are dsesigned for operation between 4 and 30 cm. The give an analog voltage corresponding to the distance from an object. Figure 3 shows the response of the sensors. I obtained this data experimentally. The curve is not linear, but if linear output is desired, a linearizing function can be implemented in software using the collected data.

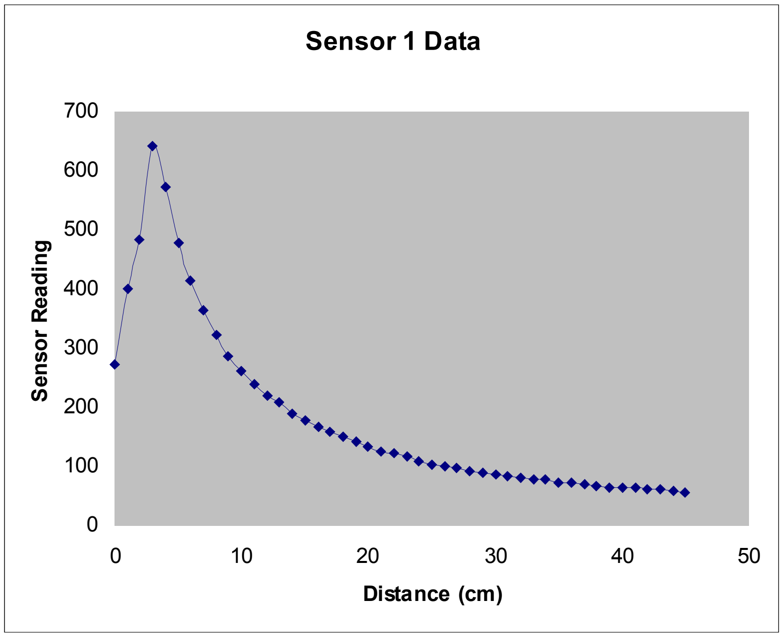

Figure 3 – IR Sensor Response

I used the IR sensors for two different purposes. The first behavior I developed with I was obstacle avoidance. The IR detects when an object is close so the robot can steer around it. The second function is wall following. I used 3 IR sensors mounted in a straight line parallel to the wall to be followed. Comparing the readings of the sensors can give an angle relative to the wall. Using this information as feedback to the servos can keep the robot following at a specified distance from the wall. Code can be found in the appendix. I found working with the IR to be straightforward. No additional circuitry was required to make it work. It just needed to be hooked up to the A/D port and it was ready.

I programmed Atlas to self-calibrate the IR sensor for obstacle detection at startup. When Atlas is reset, the first thing it does is read the IR sensor and uses the current reading as the threshold. So Atlas needs to be turned on next to a wall or object that is the right distance away from it.

The next major sensor I used was the Devantech CMPS03 Electronic Compass. It uses the I^2C bus to transmit information, which I also found easy to use after the initial learning curve. It gives a heading in tenths of a degree precision. The compass needed to be calibrated. This just involved grounding a pin when the module was facing each of the four points on the compass. This calibrates the compass for the local magnetic field. I found that the compass is slightly suspect to interference from nearby devices, but it still gives a good general heading. I am averaging the heading anyways, which is a low pass filter, so any local spikes along a path should be smoothed out. No additional circuitry was needed to interface this module either. Code can be found in the appendix.

Although not on the robot at the moment, I am adding a bump sensor before Media Day to sense collisions.

Not technically a sensor, the RF unit I am using is the Laipac TLPA 433 transmitter/receiver pair. It is a uni-directional serial link. The transmitter is hooked to the USART of the ATMega wihle the receiver is hooked to a level shifter which is connected to the PC. While the unit works alright at my house, in the lab I get very poor results. The units are very prone to interference. I am using a Manchester encoding to ensure that the signal's average DC component does not change. The results are still poor though. Even with the transmitter and receiver right next to each other at low data rates, I only receive 1 out of every 4 packets. I have a transceiver from Laipac that is of much higher quality arriving in the mail. Hopefully I can replace the transmitter before Media Day.

Behaviors

The primary behavior of Atlas is wall following. Upon finding a wall, Atlas will always move in a forward direction and follow it in a counter-clockwise direction. This means the wall will always be on the right-hand side of the robot.. The computer will plot Atlas' coordinates until the user stops it.

A related behavior, obstacle avoidance, is also be implemented. Obstacle avoidance is integrated into the wall-following behavior and mainly acts to detect turns in the wall. Collision detection will also be a behavior on the robot.

I programmed one of the timers on my board to fire an interrupt every 200 ms. The interrupt handler reads the current heading and position and transmits the data across the wireless link. If the robot is turning, it transmits a code telling the computer about its state. Figure 2 above shows the flowchart that Atlas' behaviors follow.

Experimental Layout and Results

I assembled and tested every component and subsystem of my robot individually before incorporating them into the design. I started with my main board. I tested every header and made sure it was soundly connected to the correct pin. I then ran tests on the board to ensure correct operation.

Next, I learned how to interface an LCD. Since I was going to have a wireless link, I replaced the LCD interface with a serial connection. Much more debugging information can be output that way too. The serial port was a valuable debugging tool. I characterized each of my sensors experimentally. I recorded distance vs. voltage for every IR sensor. I recorded sample compass readings and compared them to the expected reading.

I tested the servos to find the neutral points of each. I found it much easier taking it piece by piece and individually testing components. This made mistakes much easier to spot. After testing them individually, I started to integrate the sensors. Finally I started writing the computer program to capture heading and position value, which worked great.

Although the step-by-step approach helped a lot, I found that what hurt me was disorganization. I had wires going everywhere, which caused confusion and unreliability. Instead of making proper connectors, I would have alligator clips or some other unreliable connection which would lead to errors that were frustrating. Also, there were times when I threw a lot of stuff together all at once, and that caused me trouble also.

Conclusion

Atlas is an autonomous mobile robot that uses its sensors to achieve its purpose and goal. It can collect data about a room and make a map. Each sensor and system is vital to how Atlas models its surroundings and communicates with the computer. It was important to integrate them correctly to produce the correct data and behavior patterns.

I enjoyed this class very much and the information I have learned this semester is invaluable. Building Atlas taught me a lot about design methodology. I learned a lot of what to do, and a lot of what not to do. I enjoyed watching all of Atlas' sensors and systems slowly come together to form a complete design.

Appendix

Code