SwampBoard.com Bot

Foley Ma
Spring 2006 IMDL

Table of Contents

Title Page	Page 1
Table of Contents	Page 2
Abstract	Page 3
Executive Summary	Page 4
Introduction	Page 5
Integrated Systems	Page 6..7
Mobile Platform	Page 8
Actuation	Page 9
Sensors	Page 9..14
Behaviors	Page 14
Experimental Layout and Results	Page 14
Documentation	Page 15
Appendix	Page 16..36

Abstract

The purpose of the SwampBoard.com Bot is to help students get more money for their books. It acts as a book repository system for students who can not meet their buyers or seller due to schedule conflicts. This prototype system communications with the computer via Bluetooth to receive orders and follows them as necessary. It can sense if there's open shelf space, or whether a book is present on the bookshelf. Using RFID the robot will sense which book it currently is looking at and pick it up if it is correct.

Executive Summary

The robot, named SwampBoard.com uses actuation and four sensors in order to facilitate the transaction of used books between UF students. The actuation it uses consists of two servos driven by pulse width modulation. Two other servos are mounted on the robot in order to retrieve the books. The four sensors it uses are as follows: RFID, Bluetooth, line tracker, and IR depth sensor.

The RFID sensor is used to determine which book is being sold and which book is being retrieved. Bluetooth is used to setup a link between a computer and the laptop. The line tracker is used to keep the robot following the track that goes between the user and the book shelf. The IR depth sensor is used to determine whether or not a book is on the shelf. With this information the robot would know whether or not to scan the book, place the book on the shelf or move on.

Two bodies were created for the robot. One was a prototype that allowed for full integration of the electronics. A second body was made that was more appealing to the eye. Two bookshelves were made, following the same design pattern as the robot body. The first bookshelf, a prototype, demonstrated the ability to hold books for the robot. The second shelf is sturdier and appealing to the eye.

Introduction

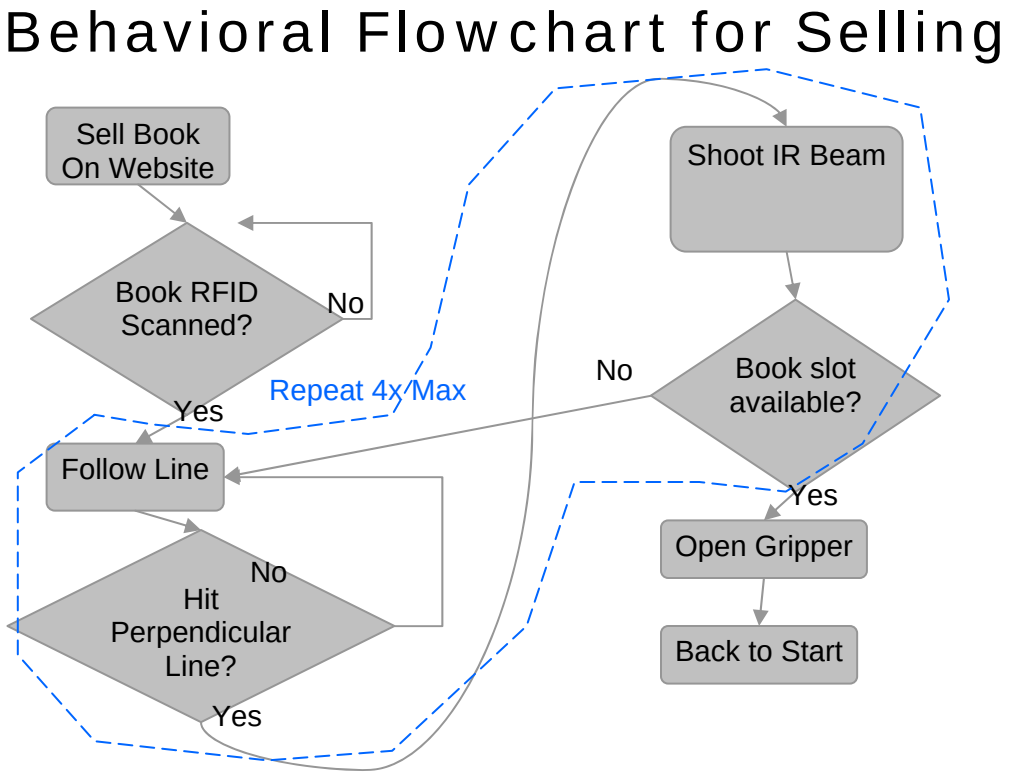
SwampBoard.com is a free service I started in Fall, 2005 at UF. The basic idea is to facilitate student to student textbook buying and selling through the Internet. If the seller of a book does not know of a student to sell the book to, the seller would go onto SwampBoard and post the book for sale. A buyer in need of a book would go on SwampBoard, search for the book, and contact the seller. The buyer and seller proceed to meet on campus and exchange the money and book in person.

The problem arises when the seller and buyer have a conflict of schedule and find it hard to meet on campus. This is where SwampBoard.com Bot comes into play. It is an automated book repository system. The seller would give the robot the book and the robot would place it on the shelf and dispenses money. When the appropriate buyer comes, the robot retrieves the appropriate book.

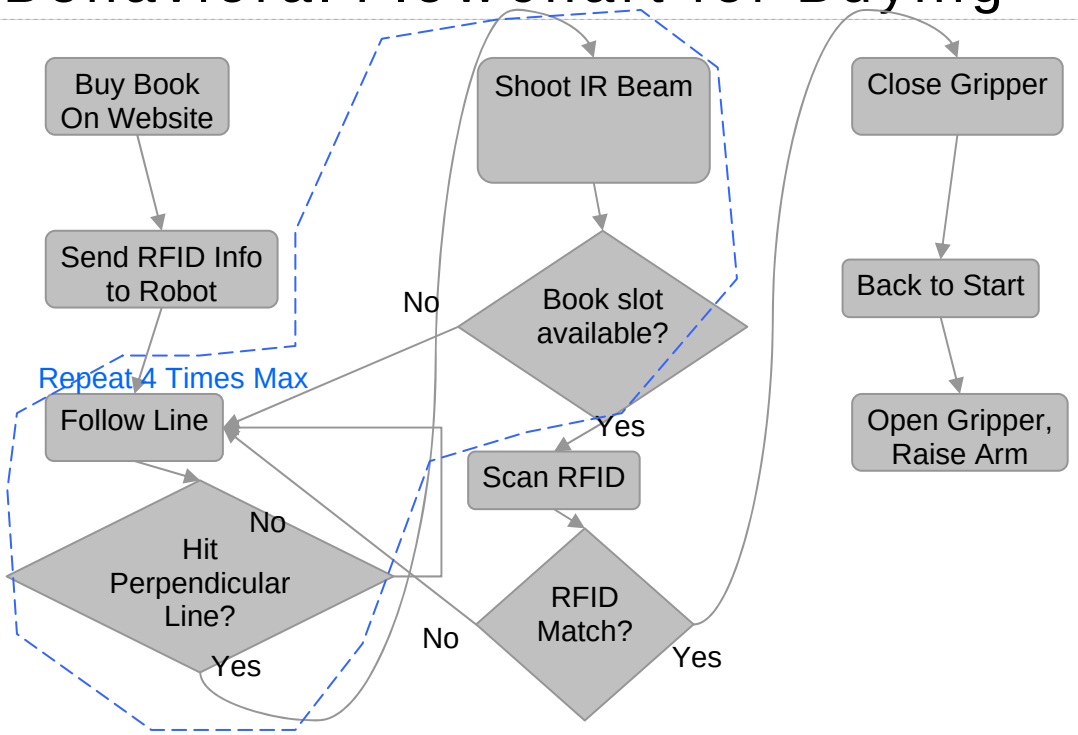
The system to be developed in IMDL is a prototype and is only to demonstrate the feasibility of such a robot. It will not be fully integrated to the Internet and will have devices in place to help the robot complete its task – devices which would not be fully present in a real system.

The next several pages will describe the integration of the system, mobile platform, actuation, sensors used, and what kind of behaviors go with the sensors. In the end, I will present the results as well as a conclusion

Flowchart of Operation

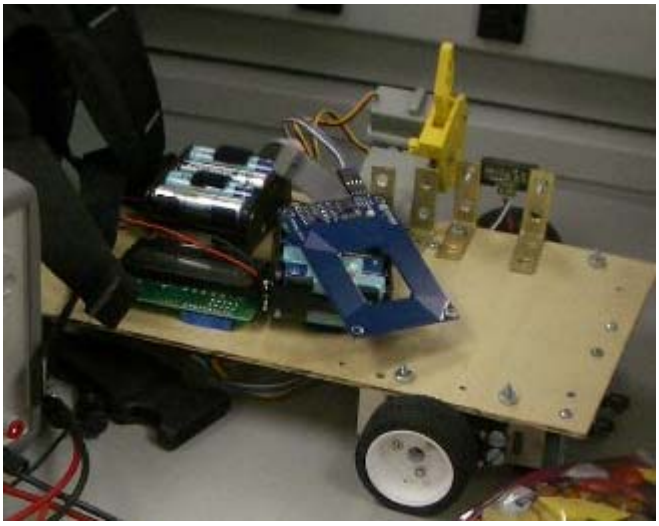


Behavioral Flowchart for Buying



Mobile Platform:

There are two platforms for the robot. One is a prototype, the other the final.



Prototype

The prototype is more or less, actuation on a piece of board. Very little time was spent on the consideration of the prototype. The prototype was made entirely from a Dremel drill. There are many holes on the board from mistakes. The powerboard and Mavric board are located beneath the platform to hide the wires. The gripper is located on the left of the platform with the IR depth sensor besides it. Both are mounted with 90^0 brackets. In the front of the platform are three IR sensors for line detecting. They are lowered towards the ground with long screws.

Actuation:

There are four servos on the robot. Two of them are for motion, with one driving the left wheel, the other the right wheel. The remaining two servos are for the arm, one controlling the elbow the other the gripper. The gripper moves vertically, grasping books that are laid down. The elbow moves vertically to assist the gripper in removing and putting down books.

Sensors:

The next section will describe the Bluetooth module, RFID reader, line tracker and depth sensor.

Bluetooth Module:

The Bluetooth module used on the SwampBoard.com bot is used to communicate with the server. The module works as a serial pipe between the robot and server and runs at 2.4~2.54Ghz spectrum. It handles a bit rate between 9600 and 115200. The advertised range for the unit is 350 feet.

Using the Bluetooth module requires the use of a Bluetooth receive on the server. This may be built into a laptop, or come in the form of a Bluetooth dongle. In my case I am using a dongle. When starting communication between the module and dongle, access must be granted by the server. Once access is granted, the server will remember that authorization has been granted by the unique identification on the module.

Once authorization has been granted, a communication port must be open. By default a communication port is not open. Two communication ports may be open. Windows calls it incoming and outgoing and it will assign two different port numbers. From my experience, only the outgoing is necessary. Using the outgoing port will allow a terminal program to receive and transmit data.

To communicate to the robot on the server side, any terminal program that can access the communication port will do. HyperTerminal and Gray's Terminal were used for testing. Gray's Terminal is recommended because it has the option to display data in binary and hex format, rather than just ASCII.

To communicate to the server on the robot side, a UART is used. The Bluetooth module has four pins, power, ground, transmit and receive.

Figure 1.

The receive on the Bluetooth hooks up to my PORT E1, which is transmit on the Mavrick Board. The transmit on the Bluetooth is hooked up to PORT E0, which is receive.

The Bluetooth will be accessed via a web based application. The programming will be done in PHP and HTML. A serial port extension was downloaded from http://www.easyvitools.com/

RFID Reader:

The RFID reader was used to identify each unique book in the collection.

It communicates at a 2400 bps, 8 bits, 1 stop, no parity. It contains four pins: power, ground, sout and enable. The power required is +5V DC. The RFID reader will be enabled when the enable pin is low. Sout transmit data at TTL levels.

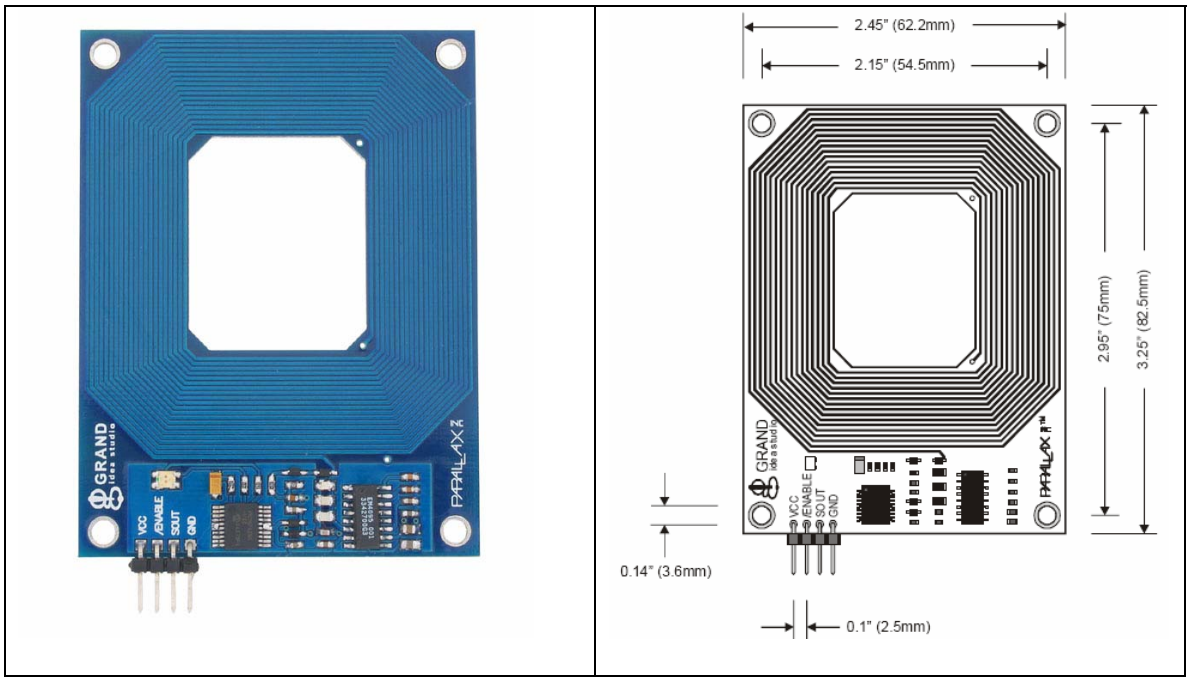


Figure 2

The purpose of the RFID Reader for SwampBoard.com bot is to identify each of the different books. The RFID Reader requires an RFID tag attached to the book. The tags must be parallel to the reader when being read. See Figure 3 for the dimensions of the tag.

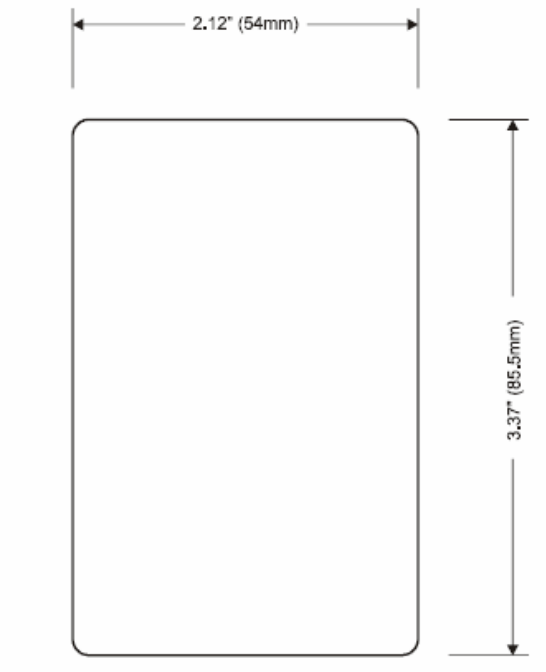


Figure 3

The Parallax RFID Reader Module works exclusively with the EM Microelectronics-Marin SA EM4100-family of passive read-only transponder tags. A variety of different tag types and styles exist with the most popular made available from Parallax. Each transponder tag contains a unique identifier (one of 240, or 1,099,511,627,776, possible combinations) that is read by the RFID Reader Module and transmitted to the host via a simple serial interface. Each ID transmits a 12 bit code. 10 of which are data. See figure 4 for a graphical display of the data bits.

MSB LSB

Start Byte (0x0A)	Unique ID Digit 1	Unique ID Digit 2	Unique ID Digit 3	Unique ID Digit 4	Unique ID Digit 5	Unique ID Digit 6	Unique ID Digit 7	Unique ID Digit 8	Unique ID Digit 9	Unique ID Digit 10	Stop Byte (0x0D)

Figure 4

The RFID tag can be read up to 3” away and as close as physical contact with the RFID reader. The reader strength is dependent on the voltage going into the RFID. The sensitivity decreases as voltage drops.

Line Tracker:

The line tracking device is called QTI. It was originally built for a robot named QTI on www.Parallax.com The sensor contains a IR module, 2 resistors and one capacitors. It shoots an IR beam, and depending on how much is received back into the detector, it emits an analog signal from 0 to about 2.8V.

Sharp GP2Y0A21YK Depth Sensor

The Sharp depth sensor was purchased from Sparkfun. It is similar to the Line Tracker where it shoots out a IR beam and outputs an analog signal depending on how much IR is received. The IR it receives is reflected off of objects that are near the proximity. The range of the analog output is from 0 to 3.3V.The sensor was used to determine if a book is on the bookshelf. If a book is present the depth will be smaller than that of when there is no book.

Behaviors

The robot has two main behaviors: selling and buying.

When a user is selling a book, the robot will be activated via a Bluetooth signal. It will wait for a RFID tag to be read. Once read, the robot will grasp the book and place it on an empty bookshelf. The robot will determine whether a bookshelf is empty or not by using the depth sensor. Once the book has been placed the robot goes back home

When a user is buying a book, the robot will be activated via a Bluetooth signal. The signal contains the RFID number selected by the user. With the RFID number in hand, the robot will scan every available book on the shelf until an ID matches. Once the ID matches the robot will retrieve it and head home.

Experimental Layout and Results

The threshold for the line tracker is set at 1.6V. If it goes below this, then the robot is on white.

The threshold for the depth sensor is set at 1V. If it is greater than this then there is a book at the position. If it's less than this, it is assumed the book is not there.

Conclusion

Overall, this was a pretty intense project. I've achieved everything I wanted to, except, I was unable to implement a money reader. Although the money reader is not essential to the project, it would have been a nice sensor to have. In the future I may implement it, since I already have the necessary hardware to integrate with it and know how it works. I also did not have time to write more database code. For example, if you buy a book from the website, it will not remove the book from the list. It does not represent a realistic environment, but it is good enough for a prototype.

I am fairly happy that I was able to integrate a lot of communication functions. This includes using Bluetooth and rfid. The Bluetooth impressed me the most and I will do more development with it in the future.

For students of the future, I would recommend that they plan out the body of the robot. I am rather short sighted in this area and would rather build the robot by hand rather than use autoCAD. The advantage of building two bodies, one being a really ugly prototype is that when you spend time on the final body, you know more of what you want to build and it looks nicer.

Future work on the robot include a better arm. The current arm looks too much of a toy, and is rather short.

Documentation

AVR Freaks

www.avrfreaks.net

Sparkfun Electronics

www.sparkfun.com

BDMicro Mavric128 Development Board

www.bdmicro.com

Parallax, Inc.

www.parallax.com

Serial Port and Communications Made Easy

http://www.easyvitools.com/

Appendices

Code on robot

```
        #include <avr/io.h>
#include <avr/interrupt.h>
//#include <avr/signal.h>
#include <util/delay.h>
#include <inttypes.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>
#include <avr/pgmspace.h>
#define RX_COMPLETE (1<<RXC)

#include <inttypes.h>
#define I2C_START (_BV(TWINT)|_BV(TWSTA)|_BV(TWEN))
#define I2C_MASTER_TX (_BV(TWINT)|_BV(TWEN))
#define I2C_TIMEOUT 50000 //1000
#define I2C_ACK 1
#define I2C_NACK 0
#define SERVO1 OCR1A
#define SERVO2 OCR1B
#define SERVO3 OCR1C
#define FF 0
#define LT 1
#define RT 2
//#define SPEED_MIN 476
#define SPEED_MIN 948
#define SPEED_MIN2 1286
#define SPEED_MID 1324
#define SPEED_MAX2 1362
#define SPEED_MAX 1500
#define Forward_left 1500
#define Forward_right 1100
//#define SPEED_MAX 1800

#define SPEED_stop 0
#define ADC_VREF_TYPE 0x00

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
```

```
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

int read_adc(unsigned char adc_input);
void init_servos(void);
void motor_forward(void);
void motor_stop(void);
void turn_right(void);
void turn_left(void);
void start_ad(int channel);
void left_QTI(void);
void right_QTI(void);
void init_ad(void);
void adc_start(void);
uint16_t adc_read(void);
void adc_wait(void);
void adc_chsel(uint8_t channel);
void USART_Init( void );
char USART1_Receive( void );
void USART1_Transmit(char transmitchar);
char USART0_Receive( void );
void USART0_Transmit(char transmitchar);
void initlcd(void);
void writecommand(int data);
void writedata(int data);
void writeletter(char letter);
void fivedelay(void);
void writestring(char []);
void init_servos(void);
void arm_up(void);
void arm_off(void);
void arm_down(void);
void gripper_off(void);
void gripper_open(void);
void gripper_close(void);
void linetrack(void);
void readRFID(char * RFID);
int bookonshelf(void);
void getoffline(void);
void clearscreen(void);
void nobookonshelf(void);
void yesbookonshelf(void);
uint16_t adc_readn(uint8_t channel, uint8_t n);

#include <util/delay.h>


int main(void)
{
```

```
init_ad();
USART_Init();
DDRE = 0x18;

initlcd();
init_servos();

gripper_open();
arm_up();

char ReceivedChar;

//USART1 is the RFID USAT0 is Bluetooth
char RFID[11];

while(1)
{
	gripper_open();

	ReceivedChar = USART0_Receive();
	writeletter(ReceivedChar);

	if(ReceivedChar == 'B')
	{
		int shelfnumber = 0;
		char chardepth[10];
		int depth = 0;
		char tempRFID[11];
		char buyRFID[11];
		for(int i = 0; i < 11; i++)
		{
			buyRFID[i] = USART0_Receive();
		}
		buyRFID[10] = '\0';
		writestring(buyRFID);
		arm_down();

		for(int i = 0; i < 300; i++ )
		{
			_delay_loop_2(30000);
		}


                                                        //find the book now

		for (int i = 1; i < 5; i++)
		{
			arm_off();
			gripper_off();
			getoffline();
			linetrack();
```

```
                depth = bookonshelf();
                itoa(depth, chardepth, 10);
                clearscreen();
                writestring("TRACK");
                itoa(i, chardepth, 10);
                writestring(chardepth);

                depth = bookonshelf();                  //need to read
twice for some reason
                itoa(depth, chardepth, 10);
                writestring(chardepth);
                shelfnumber = i;            //save position

                if(depth < 200)
                {
                        writestring("NOBOOK");          //go to
next shelf
                        getoffline();

                }
                else
                {
                    writestring("BOOK");

                    readRFID(tempRFID);
                    writestring(tempRFID);
                    if(strcmp(tempRFID,buyRFID)==0)
                    {
                        gripper_close();
                        for(int i = 0; i < 30; i++ )
                        {
                            _delay_loop_2(30000);
                        }
                        arm_up();
                        i = 5;                          //get
out of loop
                    }
                    else
                    {
                    writestring("NOMATCH");
                    getoffline();
                    }
                }
            }
                        //Close gripper
            for(int i = 0; i < 300; i++ )
            {
                _delay_loop_2(30000);
            }
            arm_off();
            for (int i = shelfnumber; i < 5; i++)       //is it 4??
```

```
		{

			clearscreen();
			writestring("HOME");
			itoa(i, chardepth, 10);
			writestring(chardepth);
			getoffline();
			linetrack();		//Go back home

		}
	}

	if(ReceivedChar == 'D')
	{
		int depth = 0x333;
		writestring("START");
		char chardepth[10];

		while(1)
		{

			adc_chsel(0);
			adc_start();
			adc_wait();
			depth = adc_readn(0,10);
									//10 bits is 5V
									//if my depth
is
			itoa(depth, chardepth, 10);
			writestring(chardepth);

			for(int i = 0; i < 300; i++ )
			{
				_delay_loop_2(30000);
			}

			clearscreen();
		}
	}
	if(ReceivedChar == 'S')
	{							//User is selling a
book. Read RFID, TRansmit it to computer. Grip the book. Move to first black line
		char RFID[11];
		readRFID(RFID);
		clearscreen();
		//writestring(RFID);
		gripper_close();
		char chardepth[10];
		int depth = 0;
		int shelfnumber = 0;
		for(int i = 0; i < 30; i++ )
```

```
                {
                        _delay_loop_2(30000);
                }
                                                                        //wait
until gripper is close before moving on
                getoffline();
                for (int i = 1; i < 5; i++)
                {
                        linetrack();
                        depth = bookonshelf();
                for(int i = 0; i < 30; i++ )
                {
                        _delay_loop_2(30000);
                }
                        depth = bookonshelf();

                        depth = bookonshelf();                  //need to read
twice for some reason
                        itoa(depth, chardepth, 10);
                        clearscreen();
                        writestring("DEPTH");
                        writestring(chardepth);
                        writestring("TRACK");
                        itoa(i, chardepth, 10);
                        writestring(chardepth);
                        if(depth < 200)
                        {
                                nobookonshelf();
                                shelfnumber = i;        //know how far you have
gone
                                i = 5;                  //get out of this loop
                        }
                        else
                        {
                                yesbookonshelf();
                        }

                }

                for (int i = shelfnumber; i < 5; i++)
                {
                        clearscreen();
                        writestring("HOME");
                        itoa(i, chardepth, 10);
                        writestring(chardepth);
                        getoffline();
                        linetrack();            //Go back home

                }

        gripper_open();
```

```
                arm_up();
                for(int i = 0; i < 30; i++ )
                {
                        _delay_loop_2(30000);
                }
                                        for(int i = 0; i < 30; i++ )
                {
                        _delay_loop_2(30000);
                }
                gripper_off();
                arm_off();
                writestring("DONESELLING");
                }

                if(ReceivedChar == 'R')
                {
                        char tempRFID[11];
                        readRFID(tempRFID);
                        writestring(tempRFID);
                        int i;
                        for(i = 0; i < 10; i++)
                        {
                                USART0_Transmit(tempRFID[i]);
                        }
                }

        }               //end of while(1)
}                       //end of main

void init_ad(void)
{
  /* configure ADC port (PORTF) as input */
  DDRF  = 0x00;
  PORTF = 0x00;

  ADMUX = _BV(REFS0);
  ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR) |
_BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0);
}

/*
 * adc_chsel() - A/D Channel Select
 *
 * Select the specified A/D channel for the next conversion
 */
void adc_chsel(uint8_t channel)
{
  /* select channel */
  ADMUX = (ADMUX & 0xe0) | (channel & 0x07);
}
```

```
/*
 * adc_wait() - A/D Wait for conversion
 *
 * Wait for conversion complete.
 */
void adc_wait(void)
{
  /* wait for last conversion to complete */
  while ((ADCSR & _BV(ADIF)) == 0)
    ;
}

/*
 * adc_start() - A/D start conversion
 *
 * Start an A/D conversion on the selected channel
 */
void adc_start(void)
{
  /* clear conversion, start another conversion */
  ADCSR |= _BV(ADIF);
}


/*
 * adc_read() - A/D Converter - read channel
 *
 * Read the currently selected A/D Converter channel.
 */
uint16_t adc_read(void)
{
  return ADC;
}

uint16_t adc_readn(uint8_t channel, uint8_t n)
{
  uint16_t t;
  uint8_t i;

  adc_chsel(channel);
  adc_start();
  adc_wait();

  adc_start();

  /* sample selected channel n times, take the average */
  t = 0;
  for (i=0; i<n; i++) {
    adc_wait();
    t += adc_read();
    adc_start();
```

```
  }

  /* return the average of n samples */
  return t / n;
}


void USART_Init(void)
{
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: Off
// USART1 Mode: Asynchronous
// USART1 Baud rate: 2400
UCSR1A=0x00;
UCSR1B=0x10;
UCSR1C=0x06;
UBRR1H=0x01;
UBRR1L=0xA0;
}


char USART1_Receive(void)
{
char status;
char data;

     while (((status=UCSR1A) & RX_COMPLETE)==0)
           ;
     data=UDR1;

           return data;
}

void USART0_Transmit(char transmitchar)
{
          while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
          UDR0=transmitchar;
```

```
}

char USART0_Receive(void)
{
char status;
char data;

    while (((status=UCSR0A) & RX_COMPLETE)==0)
          ;
    data=UDR0;

          return data;
}

void USART1_Transmit(char transmitchar)
{
        while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
        UDR1=transmitchar;
}

/*
When interrupt-driven data reception is used, the receive complete routine must read the
received data from UDR in order to clear the RXC flag;
otherwise a new interrupt will occur once the interrupt routine terminates.

*/

void writestring(char lcdstring[])
{
        int i = 0;
        while(lcdstring[i] != '\0')
        {
                writeletter(lcdstring[i]);
                i++;
        }
}

void initlcd(void)
        {
        DDRA = 0xFF;          //make POrtA all output
        DDRB = 0xFF;
        writecommand(0x30);
        writecommand(0x30);
        writecommand(0x30);
        writecommand(0x20);
        writecommand(0x20);
        writecommand(0xC0);
        writecommand(0x00);
        writecommand(0xF0);
        writecommand(0x00);
        writecommand(0x10);
```

```
	}
void writecommand(int data)
{
	PORTB ^= 0x01;    /* toggle LED */
	PORTA = data;
	fivedelay(); //wait (10 * 120000) cycles = wait 1200000 cycles
	PORTA |= 0x08;		//Enable true
	fivedelay();
	PORTA &= 0xF7;		//disable enable
	fivedelay();
	return;
}

void writedata(int data)
{
	PORTB ^= 0x01;    /* toggle LED */
	PORTA = data;
	fivedelay();
	PORTA |= 0x0A;		//Enable and RS
	fivedelay();
	PORTA &= 0xF2;
	fivedelay();
	return;
}

void clearscreen(void)
{
	writecommand(0x00);
	writecommand(0x10);
	return;
}

void writeletter(char letter)
{
	switch (letter)
	{
	case '*':
			writedata(0xF0);
			writedata(0xE0);
			break;
		case ' ':
			writedata(0x20);
			writedata(0x00);
			break;
		case '0':
			writedata(0x30);
			writedata(0x00);
			break;
		case '1':
			writedata(0x30);
			writedata(0x10);
```

```
            break;
    case '2':
            writedata(0x30);
            writedata(0x20);
            break;
    case '3':
            writedata(0x30);
            writedata(0x30);
            break;
    case '4':
            writedata(0x30);
            writedata(0x40);
            break;
    case '5':
            writedata(0x30);
            writedata(0x50);
            break;
    case '6':
            writedata(0x30);
            writedata(0x60);
            break;
    case '7':
            writedata(0x30);
            writedata(0x70);
            break;
    case '8':
            writedata(0x30);
            writedata(0x80);
            break;
    case '9':
            writedata(0x30);
            writedata(0x90);
            break;
    case 'A':
            writedata(0x40);
            writedata(0x10);
            break;
    case 'B':
            writedata(0x40);
            writedata(0x20);
            break;
    case 'C':
            writedata(0x40);
            writedata(0x30);
            break;
    case 'D':
            writedata(0x40);
            writedata(0x40);
            break;
    case 'E':
            writedata(0x40);
```

```
            writedata(0x50);
            break;
case 'F':
            writedata(0x40);
            writedata(0x60);
            break;
case 'G':
            writedata(0x40);
            writedata(0x70);
            break;
case 'H':
            writedata(0x40);
            writedata(0x80);
            break;
case 'I':
            writedata(0x40);
            writedata(0x90);
            break;
case 'J':
            writedata(0x40);
            writedata(0xA0);
            break;
case 'K':
            writedata(0x40);
            writedata(0xB0);
            break;
case 'L':
            writedata(0x40);
            writedata(0xC0);
            break;
case 'M':
            writedata(0x40);
            writedata(0xD0);
            break;
case 'N':
            writedata(0x40);
            writedata(0xE0);
            break;
case 'O':
            writedata(0x40);
            writedata(0xF0);
            break;
case 'P':
            writedata(0x50);
            writedata(0x00);
            break;
case 'Q':
            writedata(0x50);
            writedata(0x10);
            break;
case 'R':
```

```
                writedata(0x50);
                writedata(0x20);
                break;
        case 'S':
                writedata(0x50);
                writedata(0x30);
                break;
        case 'T':
                writedata(0x50);
                writedata(0x40);
                break;
        case 'U':
                writedata(0x50);
                writedata(0x50);
                break;
        case 'V':
                writedata(0x50);
                writedata(0x60);
                break;
        case 'W':
                writedata(0x50);
                writedata(0x70);
                break;
        case 'X':
                writedata(0x50);
                writedata(0x80);
                break;
        case 'Y':
                writedata(0x50);
                writedata(0x90);
                break;
        case 'Z':
                writedata(0x50);
                writedata(0xA0);
                break;
        default:
                break;
    }
}
void fivedelay(void)
{
    int counter = 0;
    while (counter != 1)
    {
        //wait (30000 x 4) cycles = wait 120000 cycles
        _delay_loop_2(30000);
        counter++;
    }
    return;
}
```

```
void arm_up(void)
{
OCR1B = 1550;
}

void arm_down(void)
{
OCR1B = 2300;
}

void arm_off(void)
{
OCR1B = 0;
}

void gripper_close(void)
{
//OCR1A = 1300; //1400 for almost all the way
OCR1A = 1000;
}

void gripper_open(void)
{
OCR1A = 600;
}

void gripper_off(void)
{
OCR1A = 0;
}


void linetrack(void)
{
//        DDRE = 0xFF;
//        DDRB = 0x00;
          int left_ADC = 0x333;
          int right_ADC = 0x333;
          int perpindicular_ADC = 0x333;
          int perpendicular_line = 0;

          while (perpendicular_line == 0)
          {

                    adc_chsel(1);
                    adc_start();
                    adc_wait();
                    left_ADC = adc_read();
```

```
                adc_chsel(3);
                adc_start();
                adc_wait();
                perpindicular_ADC = adc_read();

                adc_chsel(2);
                adc_start();
                adc_wait();
                right_ADC = adc_read();

                if ( perpindicular_ADC > 0x333)
                {
                motor_stop();
                perpendicular_line = 1;
                }
                else
                {
                        if ( (left_ADC > 0x333) && (right_ADC > 0x333) )
                        {
                                motor_forward();
                        }
                        else if ( left_ADC < 0x333)
                        {
                                turn_right();
                        }
                        else if (right_ADC < 0x333)
                        {
                                turn_left();
                        }
                }


        }
}

void init_servos(void)
{
        TCCR1A = 0xA0; //Clear on compare match (PORT B); PE5 = OC1A; PE6 =
OC1B
        TCCR3A = 0xA0; //Clear on compare match (PORT E); PE3 = OC3A; PE4 =
OC3B
        TCCR1B = 0x12; // CLK/8
        TCCR3B = 0x12;
        ICR1 = 20000; //50hz
        ICR3 = 2000; //1000hz
        OCR1A = 0;
        OCR1B = 0;
        OCR3A = 0;
        OCR3B = 0;
        //PORTE = 0x24; //HIGH = RIGHT : LOW = LEFT
}
```

```
void motor_forward(void)
{
        //MODE = FF;
        PORTE = 0x24;
        OCR3B = Forward_left;
        OCR3A = Forward_right;          //right motor
}

void turn_left(void)
{
        OCR3B = SPEED_stop;
        OCR3A = Forward_right;
}


void turn_right(void)
{
        OCR3B = Forward_left;
        OCR3A = SPEED_stop;
}


void motor_stop(void)
{
        PORTE = 0x24;
        OCR3A = SPEED_stop;
        OCR3B = SPEED_stop;             //right motor
}

void readRFID(char *tempRFID)
{

                int i = 0;
                DDRD = 0x08;  //POrt3 is Enable in RFID
                PORTD = 0x00;
                char ReceivedRFID = 0x0B;
                while(ReceivedRFID != 0x0A)
                {

                        ReceivedRFID = USART1_Receive();
                }
                while(ReceivedRFID != 0x0D)
                {

                        ReceivedRFID = USART1_Receive();
                        tempRFID[i] = ReceivedRFID;
                        i++;
                }
                tempRFID[10] = '\0';
                PORTD = 0x0F;
}
```

```
int bookonshelf(void)
{
//Channel 0 is IR for sure
	int depth = 0;
	depth = adc_readn(0,10);
//	adc_start();
//	adc_wait();
//	depth = adc_read();
	return depth;

}

void getoffline(void)
{
	int left_ADC = 0x333;
	int right_ADC = 0x333;

	for(int i = 0; i < 4000; i++)
	{
		adc_chsel(1);
		adc_start();
		adc_wait();
		left_ADC = adc_read();

		adc_chsel(2);
		adc_start();
		adc_wait();
		right_ADC = adc_read();

			if ( (left_ADC > 0x333) && (right_ADC > 0x333) )
			{
				motor_forward();
			}
			else if ( left_ADC < 0x333)
			{
				turn_right();
			}
			else if (right_ADC < 0x333)
			{
				turn_left();
			}
	}
}


void yesbookonshelf(void)
{
	writestring("BOOK");		//go to next shelf
	getoffline();
}
```

```
void nobookonshelf(void)
{
	writestring("NOBOOK");
	for(int i = 0; i < 30; i++ )
	{
		_delay_loop_2(30000);
	}
	arm_down();
	for(int i = 0; i < 30; i++ )
	{
		_delay_loop_2(30000);
	}
	gripper_open();
	for(int i = 0; i < 30; i++ )
	{
		_delay_loop_2(30000);
	}
	writestring("DOWNANDOPEN");
}
```

PHP/Serial Code for website

Selltorobot.php

```
<?php

		ser_open( "COM3", 9600, 8, "None", 1, "None" );
		ser_write("S");
		sleep(1);
		ser_close();

		$isbn = $_GET['id'];
		$price = $_GET['price'];
		$condition = $_GET['condition'];
		$title = $_GET['title'];
		include("config.php");
		include("opendb.php");
		echo("<br><br>");
		echo "Selling $condition $title";
		echo " for $ $price<br>";
		echo "$isbn<br><br>";


?>
Please give the robot the book<br>
```

Buyfromrobot.php

```
<?
```

```
include("config.php");
include("opendb.php");

?>
<center>


<?

$query = "Select * from books where book_isbn = '$isbn'";
$result = mysql_query($query) or die('Error, query failed. ' . mysql_error());
$numbooksselling = mysql_numrows($result);
include("closedb.php");
  if ($numbooksselling > 0)
  {
    while (($b=mysql_fetch_assoc($result)) != null)
    {
      $books[] = $b;
    }

  foreach ($books as $book)
  {
?>

You are buying this book.....<br><br><br>

            <table>
            <tr>
            <td>
                <img width = 75 height = 100 src = "<?=$book['book_jacket_url'] ?>">
            </td>
            <td width = 250>
             Title: <?=$book['book_title']?><br>
             Author: <?=$book['book_author']?><br>
             ISBN: <?=$book['book_isbn']?><br>
             </tr>
             </table>
Please pay $<?=$book['book_price']?>
  <?
  }
                ser_open( "COM3", 9600, 8, "None", 1, "None" );
                ser_write("B");
                sleep(1);
                ser_write($book['book_isbn']);
                ser_close();
  }
  ?>

</center>
```