Written Report 1

Jenna Stallings

Keybot Master

IMDL EEL 4665

Instructors: Dr. Arroyo Dr. Schwartz

TAs: Josh Weaver, Ryan Stevens, Tim Martin

Table of Contents

Abstract 3

Executive Summary 3

Introduction 3

Integrated system **3**

Mobile Platform 4

Actuation 5

Sensor 5

Behavior 8

Experimental Layout and Result 9

Conclusion 9

Documentation 9

Appendices 10

Abstract

The Keybot Master is an autonomous robot that will search for misplaced keys without the supervision of the user. This will allow for the user to tell Keybot Master to find the missing keys and then finish getting ready or gathering other items. Keybot Master will use IR signal to search for the keys. A transmitter will be placed on the key ring and act as a beacon IR signal with a certain frequency. A receiver will be placed on the robot to detect this beacon signal. When the user signals for Keybot Master to find the keys, it will travel around the room in search of the signal. Once the signal is found, the robot will home in on the keys and alert the user via flashing LEDs.

Executive Summary

The purpose of the Keybot Master is to aid users to find a set of missing keys. This is done through the use of the IR sensing. An IR transmitter was milled at the UF and placed on a key ring. This transmitter uses a modulated 38kHz signal. It acts as a beacon, continuously transmitting this signal through several IR LEDs. Three receivers are placed on the platform of the Keybot Master and used to help locate and hone in on the signal. The main receiver is placed in such a way as to detect if the signal is straight ahead. The other two signals are placed towards the side and are used to direct the platform to the right direction of the keys.

The platform of the Keybot Master is made of wood. The ATMega is mounted to the board and placed in a covered area that is able to be moved for easy access to the board. Sonar detection and bump switches are used for object avoidance. LEDs and LCD are used for notification once the keys are found.

In a manufacturing aspect, this design is completely plausible. The platform is light enough for any user to handle. The object avoidance is able to roam around any room of the house without getting stuck. The receivers are wide beam and after testing in many areas, would seem to be slightly affected by surrounding light. The transformer that is to be placed on the screen can be considerably small. The keychain can enclosed in such a way to be marketable to any type of user-male, female, young, and old. The platform can be customizable to fit any room.

Overall, the keybot Master has successfully fulfilled the proposal. The overall design can be manufactured on a mass scale.

Introduction

It is a common occurrence for human beings to misplace things. One thing that the majority of Americans use every day and misplace often, are a set of keys. When it is time to leave for work or school, the keys are missing; inevitably making the person late. If only there was something that would search for the keys, allowing the person to continue getting ready and gathering their things. Hence Keybot Master was born!

Ultimately, this would be a consumer product for household use. The transmitter would be in a keychain form and different casings would be sold to allow for users to have something to match to their personalities. The platform of the robot would also be sold in different colors to allow users to match it to the decor in their housing.

Integrated System

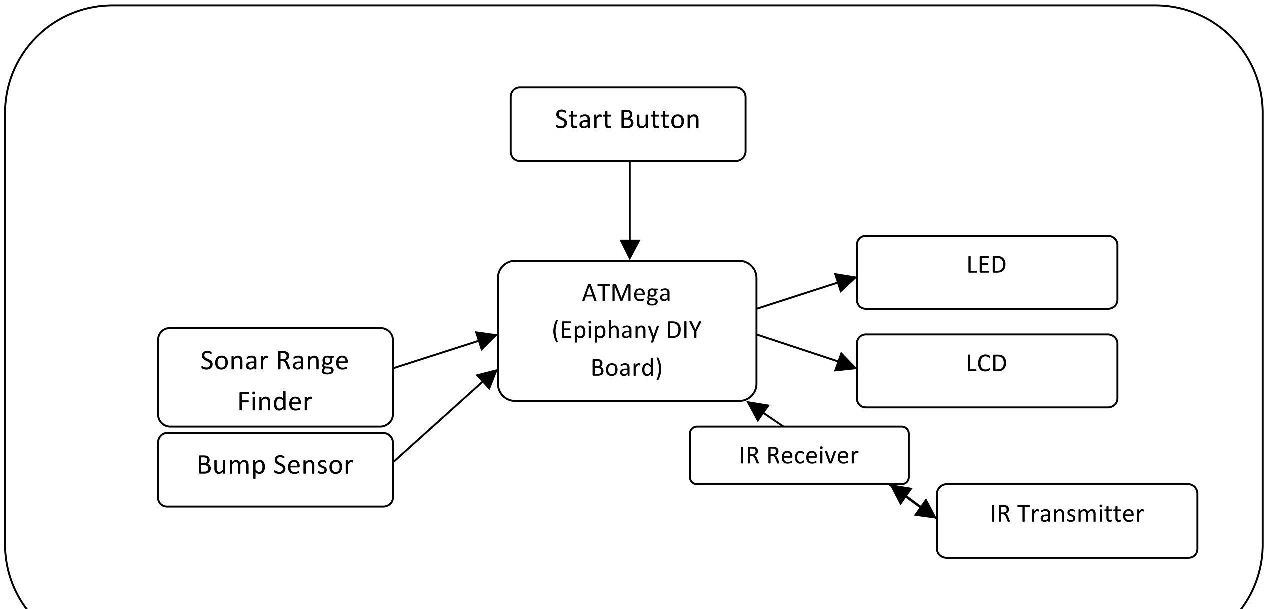

Figure 1: Flow Chart of the integrated system.

The Keybot Master is designed around the epiphany DIY board. Two Sonar Rangefinders are used as analog inputs into the ATMega used for object avoidance. The bump switches are digital signals used as a backup to the rangefinders. The IR receivers are on port pins that have edge sensing.

A transmitter PCB was milled to be a small size. This transmitter uses a 556 timer (two 555 timers) to modulate a 38kHz signal. This modulated signal is what is used to determine if the transmitter is of the right frequency.

Once the keys have been found, the LCD and LEDs are used to alert the user.

Mobile Platform

The platform is made of wood. There is the base of the platform that is circular on one side and straight edged on the other. The circular part of the base will allow for the platform to easily avoid object without getting stuck. There is a box on the bottom of the platform that houses the battery pack.

On the base there is a three wall box that covers the electronics. The top of this box is attached by hinges. This will allow for easy access to the mounted Epiphany board. On the top piece of the box, the LCD is mounted. The Sonar receivers are mounted to the front of the platform as shown in the figure below. The IR receivers are mounted on the box and tilted down slightly in order to pick up the IR transmitter on the floor. LEDs are mounted along the top of the box for indication. Bump switches are on the front and back of the platform, used for backup to the Sonar Range detector.

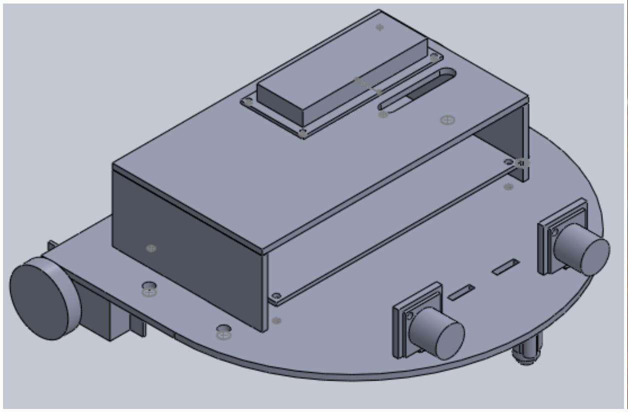
Figure 1: SolidWorks drawing of platform design.

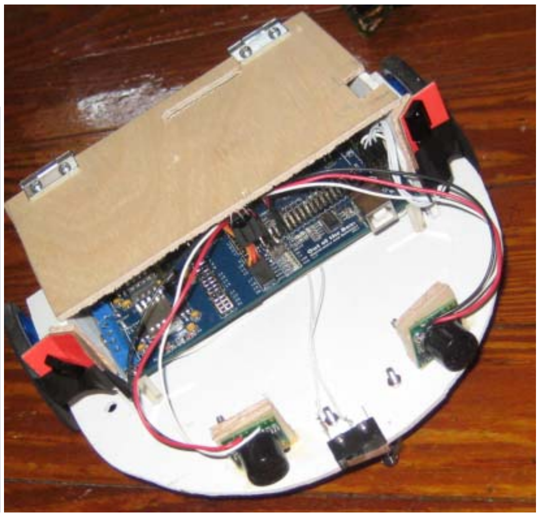
Figure 2: Photo of platform after being built.

Actuation

The platform is moved by two continuous servo motors (#948 GWS S35 STD) that were purchased from Pololu. The epiphany board has servo control already programmed into the board. When the setangle command in AVR studio is used, setting the angles to 90 will stop the servos, setting them greater than 90 will move them clockwise (in reference to the wheel side), and setting them less than 90 will move them counterclockwise. Since the wheels were mounted opposite to each other, to move the platform forward one side is set to rotate clockwise and the other is set to counterclockwise. This method is used to turn the robot right, left, and back.

Two blue silicon servo wheels (#1193 Solarbotics SW-LB) are used. They are 2-5/8" in diameter.

Figures 3 and 4: Photo of wheel and servo motor from Pololu.com reference 1.

Sensors

IR Receiver

Three IR receivers are used on the platform. The TSOP31238 receivers were chosen. They pick up a 38kHz signal that is being transmitted. These receivers are inverted; when there is no 38kHz signal the receiver is high. These receivers are used on port pins that are set to edge to detection.

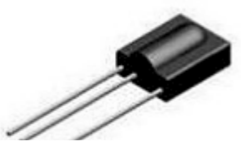
Figure 5: TSOP1238 Receiver. Photo from datasheet

IR Transmitter

The transmitter uses a single 556 timer (LM556N). The 556 timer is essentially two 555 timers. The circuit that is used is shown in the figure below. The 555 timer on the top of the figure generates the 38kHz signal. The 555 timer on the lower portion of the figure is used to modulate the signal. It generates a pulse at .108 seconds; which corresponds to a 9.25 Hz pulse. The receiver will see the pulses at 9.25 Hz.

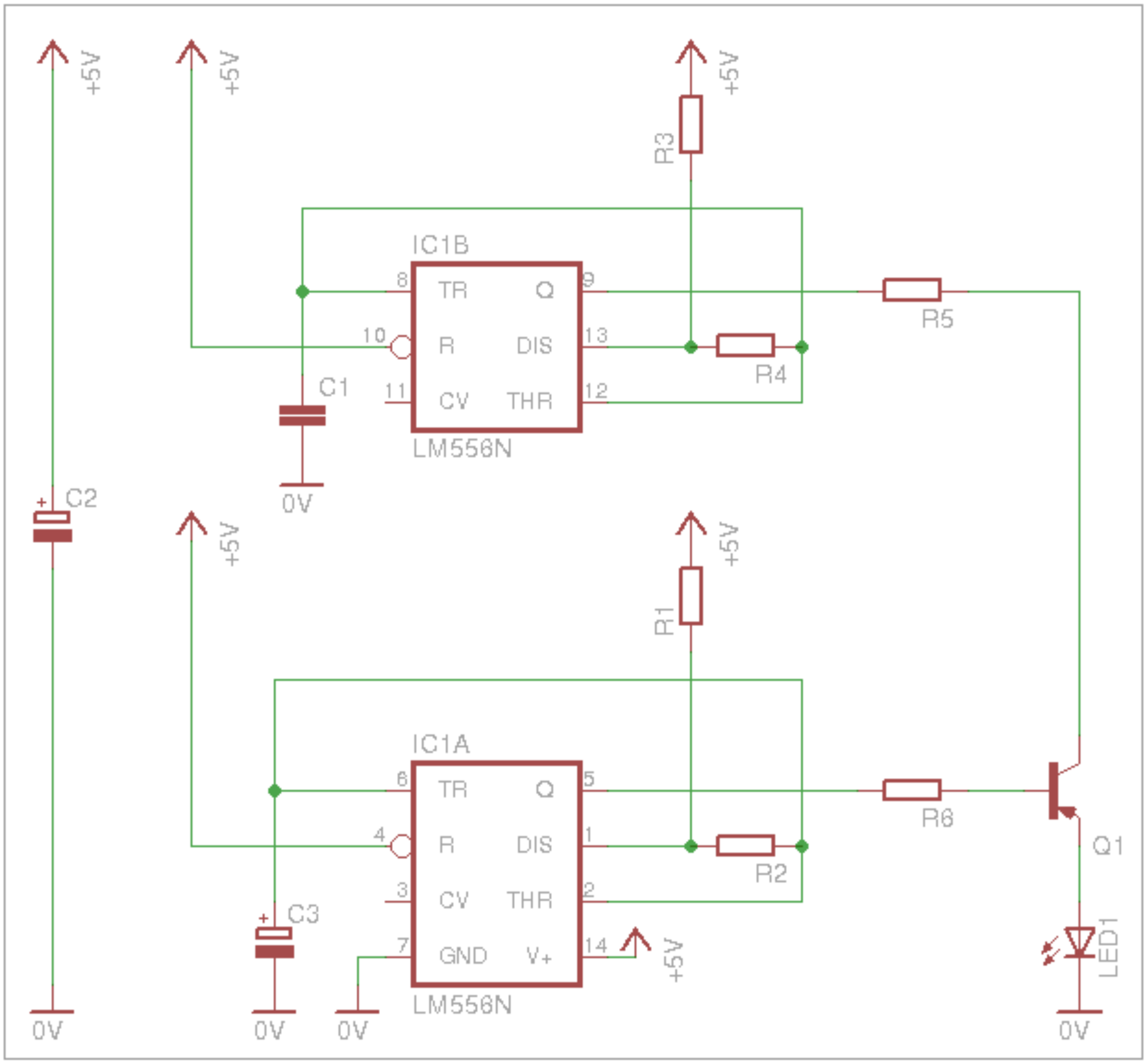

Figure 6: 556 Timer design from: http://pcbheaven.com/circuitpages/Long_Range_IR_Beam_Break_Detector/

A PCB was milled on UF's campus to implement the above circuit in a smaller capacity. The Altium design is shown in the figure below along with the actual board. The circuit was first built on a bread board and tested to make sure everything would work appropriately. Unfortunately once this board was built and populated, it was not generating the correct 38kHz signal and the receiver was not able to pick it up. A few of the capacitor values were changed and then the receiver was able to read in the signal.

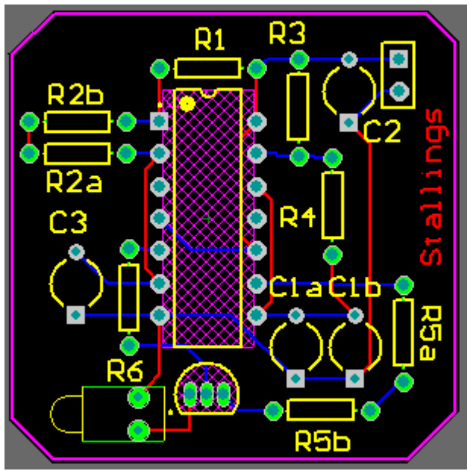

Figure 7: Altium Design of transmitter circuit

Figure 8:Final milled design of transmitter

Sonar Range Finder

Figure 9: Sonar Rangefinder from datasheet

The Maxbotix LV-EZ1 from Sparkfun is used in object avoidance. The 3.3V is used with this sensor. This sensor can detect objects between 6" and 254". Once an object is between 0 and 6 inches, the sensor will send out the corresponding 6 inch signal (this is called a dead spot). Therefore, when using this sensor if an object comes within 6 inches of the robot, the avoidance task must occur. Two sensors are hooked into the ADC of the epiphany board. The analog signal is read in, and the action is determined from the signal.

When the sensor was first hooked into the epiphany board, the analog signal that was received by the ATMega was compared to the actual distance of the object. Since the signal reads a 6 inch distance for any object that is within 6 inches, that is the threshold used for object avoidance. The corresponding analog value of 6 inches is 265. Therefore, the code used for object avoidance, reads in the analog value; if this value is less than 270, there is an object in the way and the robot must avoid it.

An IR sensor was originally going to be used to detect objects; but because an IR signal is going to be used to find the keys, the sensor has been changed to use sonar.

Bump sensor

A bump sensor is on the front and back of the robot for protection in case of an unwanted collision. This will be used as a backup to the sonar range finder. If the bump sensor bumps into something the pin goes high and the robot will move backwards or forwards (pending on which bump switch was hit) and move away from the object.

Behaviors

Obstacle Avoidance

Since there are two sonar sensors being used, the avoidance of the robot will be determined by which sensor detects the object. If the sensor on the right detects the object first the robot will turn left. If the sensor on the left detects the object fist, the robot will turn right. If both sensors detect the object at the same time (or reasonably within a certain time), the robot will back up and move around the obstacle.

Frequency capture

Originally, the frequency capture feature on the ATMega was going to be used. However, after numerous attempts at setting up the timers and events in order to do the frequency capture, the values that were being obtained were not accurate. To figure out the frequency being received, interrupts are used. Each receiver is hooked up to a port pin. The pins are set to falling edge sensing. Whenever there is a falling edge on the pin an interrupt is fired. When the interrupt is fired one of the timers is cleared. The next time there is a falling edge the timer counter is read and the frequency is determined from that value.

Hone in

There are three receivers on the platform. When the correct frequency is detected, the robot is in a certain range of the transmitter. The robot will first stop and then based on which receiver "sees" the frequency the robot will turn slightly towards the transmitter. After a certain time period the platform is in close vicinity and will alert the user.

Alert User

A counter is used based on the movements to hone in on the signal. Once the counter hits a certain number, the Robot has found the keys. When this occurs a message on the LCD screen displays that it has found keys. LEDs along the top of the robot toggle.

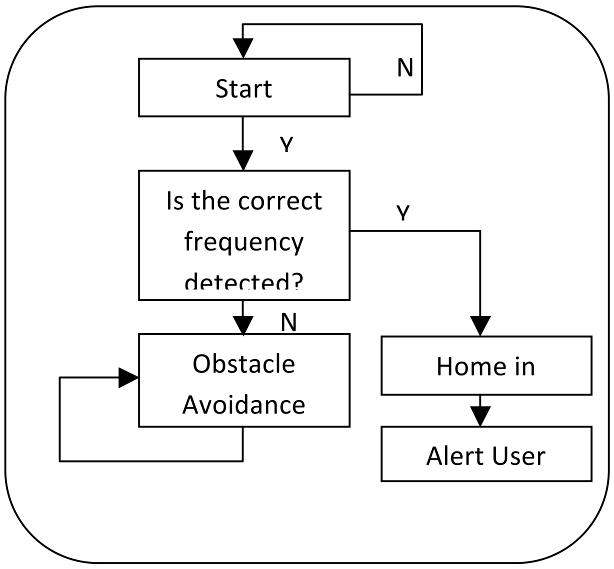

Figure 10.1: Flow chart for Main Loop

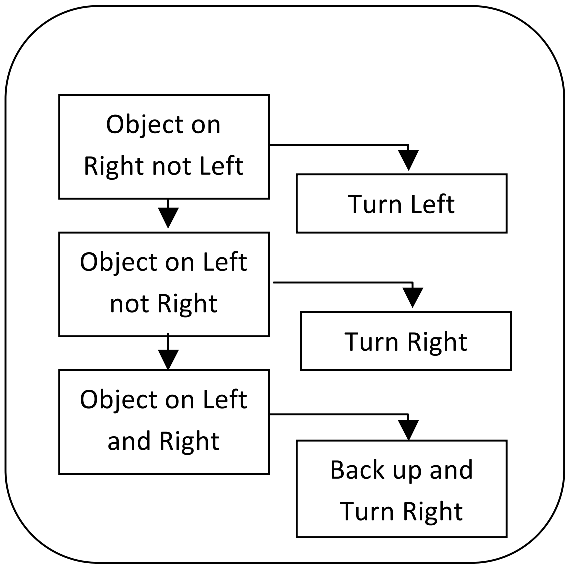

Figure 10.2: Flow chart for Obstacle Avoidance

Experimental Layout and Results

The transmitter circuit was built on a bread board to output the correct frequency IR signal. The IR receiver is then built and hooked into the epiphany board's ACD. This was used to prove that both the IR receiver and transmitter are working correctly.

The servo motors are tested by directly connecting them to the board and sending a signal. The epiphany's motor codes will be tested and tweaked from trial runs of the this initial platform. Once this bottom platform is made the finalized layout will be designed to encompass all other components (i.e. how the epiphany board will be protected in finalized use).

The rangefinder and bump sensors are tested through used of an oscilloscope. The output was measured as an object (at first a hand) moves closer to the sensors. This was used to determine if the sensor were correct in picking up an object. The sonars were then added to the epiphany board and code was written and tested to allow for the final code of object avoidance

Finally, once the board was built and the obstacle avoidance had been perfected. The way to alert the user was figured out. For testing purposes above the debug LED was used at first and then multiple LEDs were added later for final alert system.

Conclusion

Overall, there has been many issues with the original proposed design of the robot. A lot of this design complications had to do with the lack of experience in building anything mechanical. Little design flaws include where the servo motors were placed on the board; originally they were more towards the center with the idea that the overall wait should be centered. However, this design made the robot fall backwards as it moved forward. Major design flaws include where the receivers would be mounted. The original thought was that they would be mounted higher on the platform and tilted downward. However, after testing the code with this placement, the overall platform was getting in the way of the signal and the receivers had to be moved.

Overall, the Keybot Master's design has been tweaked to satisfy the original proposal. The autonomous robot can be placed on the floor and move around a room with hitting obstacles. Once the IR signal is found, the robot hones in and finds the keys. The user will be alerted and prevented from being late to work or school-inevitably saving time for the average American.

Documentation

1. "GWS S35 STD Continuous Rotation Servo", http://www.pololu.com/catalog/product/948 [4/24/12]

2. Giorgos Lazaridis, "Long Range (10mt) IR Beam Break Detector", http://pcbheaven.com/circuitpages/Long_Range_IR_Beam_Break_Detector/ [4/24/12]

3. TSOP31238 IR receiver datasheet

4. Maxbotix LV-EZ1 Sonar Rangefinder datasheet

Appendices

Program Code

```
#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"
#include <avr/interrupt.h>

#define DbLedOn()           (PORTR.OUTCLR = 0x02)             //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()          (PORTR.OUTSET = 0x02)             //Turns the debug led off.
The led is connected with inverted logic
#define DbLedToggle()   (PORTR.OUTTGL = 0x02)           //Toggles the debug led off.  The
led is connected with inverted logic
       uint16_t r1, r2, r3, timer, i,track, done;
       uint16_t frontr, leftr, rightr, frontcount, leftcount, rightcount, front, left,
right;
       uint16_t s1,s2,s3,s4,s5,s6;
       void move_forward(void);
       void move_backwards(void);
       void move_left(void);
       void move_right(void);
       void move_right_key (void);
       void move_left_key (void);
       void Int_init(void);
       void ObstacleAvoid(void);
       void foundkeys(void);
       void stop (void);
       uint16_t Leftsensor(void);
       uint16_t Rightsensor(void);
       uint16_t ReceiverF(void);
       uint16_t ReceiverL(void);
       uint16_t ReceiverR(void);

int main (void)
{
       board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                                    motorInit() is declared within because by default you
the user should define what your
                                    motor setup is to prevent hurting the Epiphany.  You
can do this by
                                  */
       RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
       DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
```

```
	ATtinyServoInit();//this function initializes the servo controller on the Attiny
	uartInit(&USARTC0,115200);//as can be seen in the schematic.  This uart is 
connected to the USB port.  This function initializes this uart
	uartInit(&USARTE1,9600);//as can be seen in the schematic.  This uart is connected 
to the Xbee port.  This function initializes this uart
	ADCsInits();//this function initializes the ADCs inside the Xmega
	LCDInit();
	Int_init();
		//portf pin 7 - start
	while((PORTF_IN & 0x80)==0){
		LCDCommand(LCD_HOME);
		fprintf(&lcd_str, "Can I help you\nfind your keys? ");
	}
	sei();
	PMIC.CTRL|=0x2;
	PMIC.CTRL |= PMIC_RREN_bm;
	uint32_t home;
	home=0;
	while (!(frontcount<4580 && frontcount>4440) && !(leftcount<4580 && 
leftcount>4440) && !(rightcount<4580 &&rightcount>4440)){
		//fprintf(&USB_str, "Avoid Obstacles: %d \n\r\n", PORTF_IN);
		LCDCommand(LCD_HOME);
		fprintf(&lcd_str, "Searching     \nAvoid Obstacles ");
		fprintf(&USB_str,"Avoid\n");
		ObstacleAvoid();
	if ((PORTF_IN & 0x80)==0x80){
		move_forward();
		_delay_ms(300);
		//fprintf(&USB_str, "back bump: %d \n\r\n", PORTF_IN & 0x08);
	}
	if ((PORTF_IN & 0x40)==0x40){
		move_backwards();
		_delay_ms(300);
	//fprintf(&USB_str, "front bump: %d \n\r\n", PORTF_IN & 0x04);
	}
		}
	while (1){
		stop();
		//_delay_ms(20);
		//move_forward();
		//_delay_ms(10);
		LCDCommand(LCD_HOME);
		fprintf(&lcd_str, "Keys are close!       \n                 ");
		if(front==1 && right==0 & left==0){
			fprintf(&USB_str, "front: %d \n\r\n", frontcount);
		move_forward();
		_delay_ms(90);
		home++;
		}
		if(front==0 && right==1 & left==0){
		move_right_key();
		_delay_ms(80);
		fprintf(&USB_str, "right: %d \n\r\n", rightcount);
		home++;
		}
		if(front==1 && right==1 & left==0){
		move_right_key();
		_delay_ms(50);
		fprintf(&USB_str, "Left: %d \n\r\n", leftcount);
		home++;
		}
		if(front==0 && right==0 & left==1){
		move_left_key();
		_delay_ms(80);
		fprintf(&USB_str, "Left: %d \n\r\n", leftcount);
		home++;
		}
		if(front==1 && right==0 & left==1){
		move_left_key();
		_delay_ms(50);
		fprintf(&USB_str, "Left: %d \n\r\n", leftcount);
		home++;
		}
		front=0;
		right=0;
		left=0;
		if(home>20){
		setServoAngle(1,90);
		setServoAngle(20,90);
		PMIC.CTRL&=0xFC;
		//fprintf(&USB_str, "found keys: %d \n\r\n", 1);
		foundkeys();
		}
		//fprintf(&USB_str, "front: %d \n %d \n %d \n\r\n",
frontcount,rightcount,leftcount);
	}
	}
void Int_init(void){
	PORTD.DIR=0xFC; //set pin 0 and 1 to input, the rest set to output
	PORTD.PIN0CTRL|=PORT_ISC_FALLING_gc; //set pin 0 to the falling edge
	PORTD.INTCTRL |=PORT_INT0LVL_LO_gc; //set in0 to med level
	PORTD.INT0MASK =0x01; //pin 0 fires int0
	PORTD.PIN1CTRL|=PORT_ISC_FALLING_gc; //set pin 1 to the falling edge
	PORTD.INTCTRL |=PORT_INT1LVL_LO_gc; //set int1 to med level
	//PORTD.INT1MASK = 0x02; //pin 1 fires int1
	TCC0.CTRLA = 0x07;
	TCC0.PER = 0xFFFF;
	TCC1.CTRLA = 0x07;
	TCC1.PER =0xFFFF;
	TCD0.CTRLA = 0x07;
	TCD0.PER = 0xFFFF;
	TCD1.CTRLA = 0x07;
	TCD1.PER =0x2FFF;
	TCD1.PER =0x2FFF;
	PORTF.DIR=0x03E; //pin 0 6 & 7 set to input
	PORTF.PIN6CTRL |=PORT_OPC_PULLDOWN_gc;// pin 6 with pullup
	PORTF.PIN7CTRL|=PORT_OPC_PULLDOWN_gc;// pin 6 with pullup
	PORTF.PIN0CTRL|=PORT_ISC_FALLING_gc; //set pin 0 to the falling edge
	PORTF.INTCTRL |=PORT_INT0LVL_LO_gc; //set in0 to med level
	PORTF.INT0MASK =0x01; //pin 0 fires int0
	TCD1.INTCTRLA =0x2;
	PORTC.DIRCLR=0x10;; //set pin 4 to input
	PORTC.PIN4CTRL|=PORT_ISC_FALLING_gc; //set pin 0 to the falling edge
	PORTC.INTCTRL |=PORT_INT0LVL_LO_gc; //set in0 to med level
	PORTC.INT0MASK =0x10; //pin 0 fires int0
}
```

```
void ObstacleAvoid(void){
                     //obstacle avoidance
              move_forward();
       if(Leftsensor()<280 & Rightsensor()<280){
              //something in front
              move_backwards();
              _delay_ms(150);
              move_right();
              _delay_ms(150);}
       if(Leftsensor()<280 & Rightsensor()>280){
              //something on left
              move_right();
              _delay_ms(300);}
       if(Leftsensor()>280 & Rightsensor()<280){
              //something on right
              move_left();
              _delay_ms(300);}
       if ((PORTF_IN & 0x80)==0x80){
              move_forward();
              _delay_ms(300);
       }
       if ((PORTF_IN & 0x60)==0x60){
              move_backwards();
              _delay_ms(300);
       }
}
void move_forward(void){
       setServoAngle(1,30);
       setServoAngle(20,120);
}
void move_backwards(void){
       setServoAngle(1,120);
       setServoAngle(20,30);
}
void move_left (void){
       setServoAngle(1,30);
       setServoAngle(20,30);
}
void move_right (void){
       setServoAngle(1,120);
       setServoAngle(20,120);
}
void move_left_key (void){
       setServoAngle(1,30);
       setServoAngle(20,90);
}
void move_right_key (void){
       setServoAngle(1,90);
       setServoAngle(20,120);
}
void stop(void){
       setServoAngle(1,90);
       setServoAngle(20,90);
}
uint16_t Leftsensor(void){
       s1=analogRead_ADCA(4);
       _delay_ms(5);
       s2=analogRead_ADCA(4);
```

```
	_delay_ms(5);
	s3=analogRead_ADCA(4);
	return (s1+s2+s3)/3;}
uint16_t Rightsensor(void){
	s4=analogRead_ADCA(1);
	_delay_ms(5);
	s5=analogRead_ADCA(1);
	_delay_ms(5);
	s6=analogRead_ADCA(1);
	return (s4+s5+s6)/3;
}
void foundkeys(){
	while(1){
	DbLedToggle();
	PORTF.OUTTGL=0x02;
	PORTD.OUTTGL=0x04;
	_delay_ms(500);
	PORTF.OUTTGL=0x02;
	PORTD.OUTTGL=0x04;
	PORTF.OUTTGL=0x04;
	PORTD.OUTTGL=0x08;
	_delay_ms(500);
	PORTF.OUTTGL=0x04;
	PORTD.OUTTGL=0x08;
	PORTF.OUTTGL=0x08;
	PORTD.OUTTGL=0x10;
	_delay_ms(500);
	PORTF.OUTTGL=0x08;
	PORTD.OUTTGL=0x10;
	LCDCommand(LCD_HOME);
	fprintf(&lcd_str, " Your keys have \n  been found!!  ");
}
}
ISR(PORTC_INT0_vect){
	if (frontr == 0){
	TCC0.CTRLFCLR |= 0x0C;// these two commands
	TCC0.CTRLFSET |= 0x08;// reset counter
	frontr = 1;
	}
	else if (frontr == 1){
	frontcount = TCC0.CNT;
	frontr = 0;
	}
	fprintf(&USB_str, "front: %d \n\r\n", frontcount);
	if((frontcount<4600 && frontcount>4200)){
		front=1;
	}

	return;
}
ISR(PORTF_INT0_vect){
//ISR(PORTD_INT1_vect){
	if (rightr == 0){
	TCC1.CTRLFCLR |= 0x0C;// these two commands
	TCC1.CTRLFSET |= 0x08;// reset counter
	rightr = 1;
	}
	else if (rightr == 1){
        rightcount = TCC1.CNT;
        rightr = 0;
        }
        fprintf(&USB_str, "right: %d \n\r\n", rightcount);
                if((rightcount<4600 &&rightcount>4200)){
                        right=1;}
                        return;
}
ISR(PORTD_INT0_vect){
        if (leftr == 0){
        TCD0.CTRLFCLR |= 0x0C;// these two commands
        TCD0.CTRLFSET |= 0x08;// reset counter
        leftr = 1;
        }
        else if (leftr == 1){
        leftcount = TCD0.CNT;
        leftr = 0;
        }
        fprintf(&USB_str, "left: %d \n\r\n", leftcount);
        if((leftcount<4600 && leftcount>4200)){
                left=1;
        }
        return;
}
ISR(TCD1_OVF_vect){
        if(track==0){
                PMIC.CTRL|=0x1;
                track=1;
        }
        else if(track==1){
                PMIC.CTRL&=0xFE;
                track=0;
        }
        fprintf(&USB_str, "home in on keys %d \n", track);
        return;
}
```