Parker Authier & Ben Kepner

Prof. Foster

CE 420-03L Lab 6

Fall Semester 11/21/2019

Contents

[Objectives 3](#_Toc26394904)

[Hardware 4](#_Toc26394905)

[Program Source Code 5](#_Toc26394906)

[Main.c 5](#_Toc26394907)

[Sonar.c 10](#_Toc26394908)

[Sonar.h 12](#_Toc26394909)

[Experiment and Data Analysis 14](#_Toc26394910)

[Conclusions 15](#_Toc26394911)

# Objectives

The objective of this lab is to use the ultrasonic sensor with the Micros II board to measure distance and display the result on an OLED display. The device should be able to be communicated to via Bluetooth terminals. The functionality and accuracy of the sensor was then compared to that of a store-bought laser distance sensor.

# Hardware

The hardware used was the Microcomputer Systems II board, the expansion board designed for the class, the LCD display module and the ultrasonic sensor module.

# Program Source Code

## Main.c

|  |
| --- |
| /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  /\*\* Provide complete documentation for your program  /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  //suppress plib warnings  #define \_SUPPRESS\_PLIB\_WARNING  #define \_DISABLE\_OPENADC10\_CONFIGPORT\_WARNING  #define CPU\_FREQ 80000000  #define ONE\_SEC\_TICK (CPU\_FREQ/2)  #include <p32xxxx.h>  #include <plib.h>  #include "sonar.h"  #include "SH1106.h"  //#include "BT\_UART.h"  // configuration bit settings, Fcy=80MHz, Fpb=40MHz  #pragma config POSCMOD=XT, FNOSC=PRIPLL  #pragma config FPLLIDIV=DIV\_2, FPLLMUL=MUL\_20, FPLLODIV=DIV\_1  #pragma config FPBDIV=DIV\_2, FWDTEN=OFF, CP=OFF, BWP=OFF  //generate time delay for the specified amount of milliseconds  void msDelay(unsigned int ms)  {  // Convert ms microseconds into how many clock ticks it will take  unsigned int ticks = ms \* ( ONE\_SEC\_TICK /1000);  // ms \*= TIMER\_FREQ / 1000000 ; // Core Timer updates every 2 ticks  \_CP0\_SET\_COUNT(0); // Set Core Timer count to 0  while (ticks > \_CP0\_GET\_COUNT()); // Wait until Core Timer count reaches the number we calculated earlier  }  void main(void){  char res[20];    INTEnableSystemMultiVectoredInt();  initSPI2(); //initialize SPI2 for OLED  initUART2(); //initialize UART2 for Bluetooth communication  initUART1();  initSonar();    //send message to the Bluetooth user  UART2\_putstr("Connection Successful\n");  UART2\_putstr("Send command to choose operating mode\n");  // UART2\_putstr("renaming bluetooth device...\n");  // UART2\_putstr("\r\nAT+NAMECHIP\_KIT5");    //initialize OLED and print sensor measurments  SH1106\_begin();  SH1106\_clear(oled\_buf);  SH1106\_string(0, 10, "Ultrasonic Sensor ", 15, 0, oled\_buf);  SH1106\_string(0, 30, "Dist: ", 15, 0, oled\_buf);  SH1106\_string(0, 50, "time: ", 15, 0, oled\_buf);  float distance = sonarDistance(); //measure distance  sprintf(res, "%5.2f cm", distance);  SH1106\_string(60, 30, res, 15, 0, oled\_buf);  SH1106\_display(oled\_buf); //display measurement on OLED    int mode = 0;  while(1)  {  char input = UART2\_getchar();    switch(input)  {  case 'R':  case 'r':  mode = 0;  UART2\_putchar(input);  if(! running)  {  running = TRUE;  triggerPulse();  }  break;    case 'P':  case 'p':  mode = 1;  UART2\_putchar(input);  running = FALSE;  break;    default:  break;  }    //UART2\_putchar(input);    switch(mode)  {  case 0:  distance = sonarDistance(); //measure distance  sprintf(res, "%5.2f cm %u ticks\n", distance, lastSonarResponceTime);  UART2\_putstr(res);  sprintf(res, "%5.2f cm", distance);  SH1106\_string(60, 30, res, 15, 0, oled\_buf);  sprintf(res, "%5d ticks", runtime);  SH1106\_string(50, 50, res, 15, 0, oled\_buf);  SH1106\_display(oled\_buf);  break;    case 1:  break;    }      msDelay(500);    }    // mIC5CaptureReady()  // mIC5ReadCapture();    //modify this main program to make the program run indefinitely  //while handling the different operating modes requested by the user    } |

## Sonar.c

|  |
| --- |
| #include "sonar.h"  // returns the distance detected by the sonar device in centimeters  float sonarDistance(void)  {  \_CP0\_SET\_COUNT(0);  unsigned int initialTime = \_CP0\_GET\_COUNT();    float returnValue = 0;  lastSonarResponceTime = secondCapture - firstCapture;  if(lastSonarResponceTime > 0x173180);  else  returnValue = (lastSonarResponceTime ) / 40.0f / 58.0f; // TODO: finish writing math to turn ticks into microseconds    runtime = \_CP0\_GET\_COUNT() - initialTime;    return returnValue;  }  void initSonar()  {  initOC5();  //initIC4();  running = TRUE;  }  void initOC5()  {  // setup OC5 with timer 2  OC5CON = 0x0000; // Turn off OC1 while doing setup.  OC5CON = 0x0004; // Configure for single pulse mode  OC5R = 0x0; // Initialize primary Compare Register  OC5RS = 0x190; // Initialize secondary Compare Register    mOC5ClearIntFlag(); // Clear the OC1 interrupt flag  mOC5IntEnable(1); // Enable OC1 interrupt  // Set OC1 interrupt priority to 7,  mOC5SetIntPriority(7); // the highest level  mOC5SetIntSubPriority(3); // Set Subpriority to 3, maximum    OC5CONSET = 0x8000; // Enable OC1      T2CON = 0x0000; // Configure Timer2 for a prescaler of 1:1    PR2 = 0x190; // Set period (PR2 is now 16-bits wide)  // set period to 10 us  // configure int  T2CONSET = 0x8000; // Enable Timer 2  }  void initIC4(void)  {  //settup input compare    IC4CON = 0b0000001100100110; // disable IC4 and configure  //IC4CON = 0b0000001000100001;  // Simple Capture Event mode ? every edge, specified edge first and every edge thereafter  // Input capture buffer is not empty  // First Capture Edge is rising  // timer3 is selected    T2CON = 0x0008; // configure timer2 for 1:1 prescaler  // 32 bit mode    PR2 = 0xffffffff;    mIC4ClearIntFlag(); // clear IC4 interrupt  mIC4IntEnable(1); // enable IC4 interrupt  mIC4SetIntPriority(7); // Set IC4 interrupt priority to 7,  mIC4SetIntSubPriority(3); // Set Subpriority to 3, maximum    IC4CONSET = 0x8000;  T2CONSET = 0x8000;  }  // Example code for Output Compare 1 ISR:  void \_\_ISR(\_OUTPUT\_COMPARE\_5\_VECTOR, ipl7SOFT) OC5\_IntHandler (void)  {  initIC4();  mOC5ClearIntFlag(); // Clear the OC5 interrupt flag  }  void \_\_ISR(\_INPUT\_CAPTURE\_4\_VECTOR, ipl7SOFT) IC4\_IntHandler (void)  {    firstCapture = IC4BUF;  secondCapture = IC4BUF;  initOC5();  mIC4ClearIntFlag(); // Clear the OC5 interrupt flag  if(running)  triggerPulse();      } |

## Sonar.h

|  |
| --- |
| 1 /\*   2 \* File: sonar.h  3 \* Author: ben  4 \*  5 \* Created on November 15, 2019, 10:41 AM  6 \*/  7 #define \_SUPPRESS\_PLIB\_WARNING  8 #define \_DISABLE\_OPENADC10\_CONFIGPORT\_WARNING  9  10 #include <p32xxxx.h> 11 #include <plib.h> 12  13 #ifndef SONAR\_H 14 #define SONAR\_H 15  16 #ifdef \_\_cplusplus 17 extern "C" { 18 #endif 19  20 #define triggerPulse() OC5CONSET = 0x8000 21  22 void initSonar(); 23  24 float sonarDistance(void); 25  26 volatile unsigned int firstCapture; 27 volatile unsigned int secondCapture; 28 volatile int running; 29 unsigned int runtime; 30  31 void initOC5(void); 32 void initIC4(void); 33  34 volatile unsigned int lastSonarResponceTime; 35  36  37  38 #ifdef \_\_cplusplus 39 } 40 #endif 41  42 #endif /\* SONAR\_H \*/ 43  44 |

# Experiment and Data Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Distance (cm) | Ultrasonic Sensor (cm) | Laser Sensor (m) | Measurement Error (%) |
| 20 | 20.00 | 0.2 | 0.00 |
| 30 | 30.13 | 0.3 | 0.43 |
| 40 | 39.51 | 0.4 | 1.24 |
| 50 | 49.93 | 0.5 | 0.14 |
| 60 | 59.44 | 0.6 | 0.94 |
| 70 | 69.29 | 0.7 | 1.02 |
| 80 | 78.5 | 0.8 | 1.91 |
| 90 | 89.18 | 0.9 | 0.92 |
| 100 | 99.32 | 1.0 | 0.68 |
| 110 | 109.96 | 1.1 | 0.04 |
| 120 | 118.42 | 1.2 | 1.33 |
| 130 | 128.52 | 1.3 | 1.15 |
| 140 | 142.32 | 1.4 | 1.63 |
| 150 | 150.23 | 1.5 | 0.15 |
| 160 | 160.50 | 1.6 | 0.31 |
| 170 | 170.17 | 1.7 | 0.10 |
| 180 | 102.21 | 1.8 | 76.11 |
| 190 | 102.28 | 1.9 | 85.76 |
| 200 | 200.31 | 2.0 | 0.15 |

Our sensor is accurate at shorter distances and becomes less so given larger distances. This is mainly because there is more interference with our ultrasonic waves as they reflect off other objects and get distorted by noise in the surrounding environment. The laser sensor is not affected by the same conditions of that of the ultrasonic sensor. Therefore, the laser sensor is more accurate at larger ranges than the ultrasonic sensor.

# Conclusions

Overall the ultrasonic sensor is more accurate at shorter ranges than longer ranges. The farther away from the ultrasonic sensor that the endpoint gets, the spread of the sound becomes larger, bouncing off more objects, and introduces more noise into the signal. The ultrasonic sensor reads the value of the largest echo/signal received, having more noise in the echo will cause the sensor to have varied readings.