**Lab 2**

Eric Pires

John Lamanuzzi

Richard Harrison

ECE 388

Team 3

Lab Date: September 21, 2016

Due Date: September 28, 2016



**Table of Contents**

[Introduction 2](#_Toc462693863)

[Methods and Testing Procedures 2](#_Toc462693864)

[Motion Sensor 2](#_Toc462693865)

[Character LCD 3](#_Toc462693866)

[Temperature Sensor 3](#_Toc462693867)

[LED 4](#_Toc462693868)

[Servo 4](#_Toc462693869)

[Results 5](#_Toc462693870)

[Motion Sensor 5](#_Toc462693871)

[Temperature Sensor and Character LCD 5](#_Toc462693872)

[LED 5](#_Toc462693873)

[Servo 6](#_Toc462693874)

[Discussion 6](#_Toc462693875)

[Conclusion 7](#_Toc462693876)

[Reflection 7](#_Toc462693877)

[References 9](#_Toc462693878)

[Appendix 10](#_Toc462693879)

Abstract

For this laboratory assignment, various sensors and actuators were tested and evaluated for the semester project. Testing procedures were designed and executed to be able to characterize and understand the sensors and actuators.

# Introduction

The objective of this lab was to determine design requirements of various sensors and actuators and their operations. Each component was researched to determine the pinout and specifications for suitable operation. The following components were utilized for this experiment:

* ATmega328p Xplained Mini board
* Hakko FX 888D
* Breadboard and Jumper Cables
* KY-033 Optical Motion Sensor
* 1602A Character LCD
* AD22100 Temperature Sensor
* KY-016 RGB LED
* SG90 Micro Servo

# Methods and Testing Procedures

## Motion Sensor

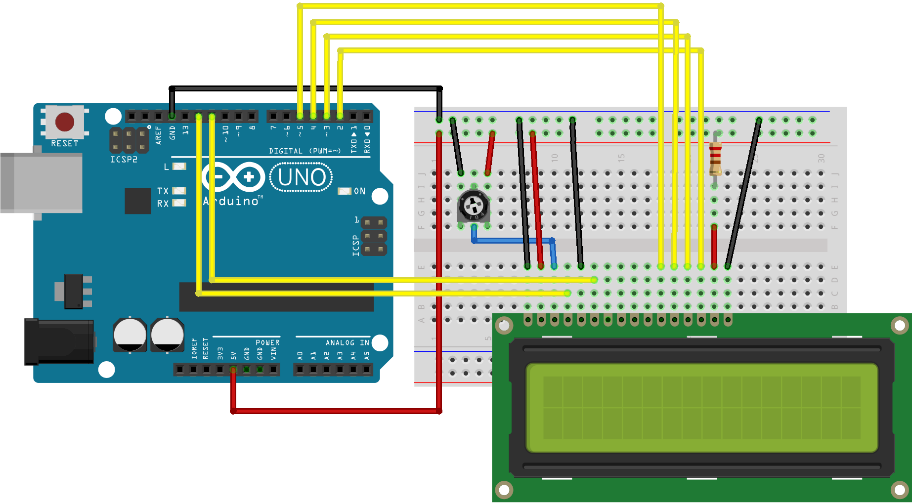
The motion sensor provided in this laboratory was the KY-033. The sensor contained 3 pins: power, ground, and output. The device was provided 5V from the microcontroller and grounded. The sensor was approached by hand to determine if it would detect motion. The sensor LED was used to indicate whether motion was present. The on-board potentiometers were adjusted to determine the length of the sensor’s detection path.

## Image result for 1602a lcd displayCharacter LCD

The 1602A LCD was provided in the laboratory. Since the component lacked breakout pins, a row of 16 male header pins was soldered onto the pads of each through-hole pin. The testing case for this device was done in conjunction with the temperature sensor. This procedure is detailed in the next section.

## Temperature Sensor

The temperature sensor provided in the laboratory was not used in this experiment. Instead, a AD22100 was tested. Based on the datasheet schematics, the device was powered 5V and grounded from left to right (Bottom View). To test this device, a simple program was written to calculate the measured temperature reading and output the value onto the character LCD provided in lab. This was written in the Arduino environment and uploaded to the microcontroller. The following pins were connected to operate the testing procedure: (LCD pin to microcontroller pin)

* VSS to GND
* VCC to 5V
* R/W to GND
* RS to PB4
* E to PB3
* D4 to PD5
* D5 to PD4
* D6 to PD3
* D7 to PD2

The output of the temperature sensor was then tied to PC0. The code can be found in Figure 1 of the Appendix.

## LED

The KY-016 RGB LED was provided in the laboratory. The sensor contained four pins: R, G, B, and GND. To test the device, the following color combinations were supplied with 5V and grounded to produce the expected color output:

* Red
* Green
* Blue
* Red and Green (Yellow)
* Blue and Green (Cyan)
* Red and Blue (Magenta)
* Red, Green, and Blue (White)

## Servo

The SG90 micro servowas provided in the laboratory. The actuator contained three pins: PWM, power, and ground. To test the device, the “Sweep” example provided in the Arduino IDE was compiled and uploaded to the microcontroller. This program was designed to rotate the shaft of the servo in a bidirectional 180-degree radius. The code for this example can be found in Figure 2.

# Results

## Motion Sensor

The motion sensor testing procedure was followed. When a hand motion was performed near the sensor’s detection path, the on-board LED was illuminated, signaling that the sensor detected the provided motion. The on-board potentiometer was adjusted to determine the range in which the sensor could detect up to. In this case, the farthest range detected was about 1-2 feet away from the sensor.

## Temperature Sensor and Character LCD

The temperature sensor and character LCD testing procedure was followed. The LCD was powered and displayed a temperature reading from within the testing environment. The measured temperature was 24 degrees Celsius (75.2 ° F). The air conditioning unit in the testing environment was set to about 79° F. Temperature data error was expected for the given device.

## LED

The RGB LED testing procedure was followed. Different colors were created following the color combinations listed in the Procedures section. A table of the various power inputs and there recorded outputs can be found below:

|  |  |
| --- | --- |
| Input | Output |
| R,GND | Red |
| G,GND | Green |
| B,GND | Blue |
| R,G,GND | Yellow |
| B,G,GND | Cyan |
| R,B,GND | Magenta |
| R,G,B,GND | White |

## Servo

The servo testing procedure was followed. The specified program and servo were uploaded and connected to the microcontroller. The shaft rotated in both directions once the signal was driven by the microcontroller.

# Discussion

As this lab dealt mostly with familiarization with possible equipment for the design project, ensuring the correct operation and interfacing was the goal. That meant understanding the pinouts, input signals, and outputs of each device, ensuring they all worked as designed.

The PIR sensor, like the servo, only had one I/O pin, meaning its output would fall on one pin (with power and ground pins), making communication to the microcontroller very simple. However, since the PIR sensor had its own LED that enabled when movement was detected, a simple flash of an LED was sufficient for determining the output of the device.

The LCD panel contained a row of pins (also with power and ground pins) that would generate a character given a valid output to the panel. The Arduino built in LCD library made the procedure simple to wire and code. The pinout found in the “HelloWorld” tutorial was used for this test, adding one addition connection to pin PC0 for the analog output of the temperature sensor.

The temperature sensor was very rudimentary; therefore, it required the use of the LCD panel to display a valuable output. The sensor had two output pins (alongside power and ground pins), and a sample block of code interfaced it with the microcontroller. Since it functions as a thermistor, it would actually return a value that must be converted using a formula provided in the datasheet. This calculation was simplified by using the function “celciusConvert” found in Figure 1. Initially, this sensor was connected incorrectly, producing an incredibly high temperature reading. Re-examining the datasheet revealed that the sensor was on backwards. Once this was fixed, the temperature reading seemed to provide accurate readings.

For the RGB LED, the pinout and outputs were simple to understand. Since all inputs would go through the R, G, B pins, and the output would be the specific color of the LED based on what pins are getting an input signal, ensuring that it would work was very simple, and a sample block of code was used to test each hue of color that the LED could produce.

The servo only contained three pins total. Interfacing with the microcontroller was simple, as only one pin was actually used for input. Loading the example code from the Arduino library was simple and easy to follow.

# Conclusion

In conclusion, this experiment provide our group with possible procedures to be implemented for the prototyping stage of our design project. Working with the various sensors and actuators helped build a better understanding of how to begin working on the project. Test-driven design was the driving factor of the lab, with each step focused on the functions of each device. This helped determine how the devices (or very similar models) could be implemented in the design project.

# Reflection

This experiment provided a useful insight for our design project. Components that would be found in a climate control and room occupancy system were tested. The functionality of each device seemed easy to grasp. Using the Arduino environment simplified the testing process. The code used in this environment can be translated into C and compiled through the Atmel Studios IDE. After testing each device, we determined that a different PIR and temperature sensor should be used in our design project. The PIR sensor used in this lab had a short range. The temperature sensor did not include a humidity sensor. These components will be replaced, but the same testing methods still apply for both.

# References

Fortier, Paul. (2016). Laboratory #2. Handout.

Mellis, David A., Limor Fred, and Tom Igoe. "Arduino - HelloWorld." Arduino - HelloWorld. Arduino, 17 Aug. 2015. Web. 26 Sept. 2016.

Schultz. Temperature Sensors TMP36 or AD22100. N.p.: n.p., n.d. INO.

# Appendix

*Figure 1: LCD and Temperature Test Plan Code*

/\*

LiquidCrystal Library - Hello World

Demonstrates the use a 16x2 LCD display. The LiquidCrystal

library works with all LCD displays that are compatible with the

Hitachi HD44780 driver. There are many of them out there, and you

can usually tell them by the 16-pin interface.

This sketch prints "Hello World!" to the LCD

and shows the time.

The circuit:

\* LCD RS pin to digital pin 12

\* LCD Enable pin to digital pin 11

\* LCD D4 pin to digital pin 5

\* LCD D5 pin to digital pin 4

\* LCD D6 pin to digital pin 3

\* LCD D7 pin to digital pin 2

\* LCD R/W pin to ground

\* LCD VSS pin to ground

\* LCD VCC pin to 5V

\* 10K resistor:

\* ends to +5V and ground

\* wiper to LCD VO pin (pin 3)

Library originally added 18 Apr 2008

by David A. Mellis

library modified 5 Jul 2009

by Limor Fried (http://www.ladyada.net)

example added 9 Jul 2009

by Tom Igoe

modified 22 Nov 2010

by Tom Igoe

modified 23 September 2016 for temperature reading

by Eric Pires

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/LiquidCrystal

\*/

// include the library code:

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int TsensorPin = A0;

double temp;

void setup() {

// set up the LCD's number of columns and rows:

lcd.begin(16, 2);

// Set sensor input to A0 (PC0)

pinMode(TsensorPin, INPUT);

// Print a message to the LCD.

lcd.print("Temp (C)");

}

void loop() {

// Calculate temperature

temp = celciusConvert(analogRead(TsensorPin));

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.setCursor(0, 1);

// Write temperature

lcd.print(temp);

// Delay sensor reading

delay(2000);

}

// Source: http://staff.mbi-berlin.de/schultz/useful\_stuff/Temperature\_sensors\_TMP36G\_and\_AD22100.ino

double celciusConvert(int raw)

{

float voltage;

voltage = raw \* (5000.0/1024); // Sensor value in mV:

return (voltage -1375) /22.5; // Temperature according to datasheet: 1.375 V <-> 0 Â°C

// 22.5 mV / Â°C; Ratiometric measurement, conversion valid for 5 V!

}

*Figure 2: Servo “Sweep” Test Code (From Arduino Examples)*

/\* Sweep

by BARRAGAN <http://barraganstudio.com>

This example code is in the public domain.

modified 8 Nov 2013

by Scott Fitzgerald

http://www.arduino.cc/en/Tutorial/Sweep

\*/

#include <Servo.h>

Servo myservo; // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

void setup() {

myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop() {

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}