Report: Drivers

Cole Rottenberg
11062528

February 26, 2024

Introduction

The overall goal of this lab was to demonstrate the capabilities of communicating of I2C to an accelerometer and with some logic controlling an 8x8 LED display. The true objective was to build a level that would be displayed on the LED display. The accelerometer would be used to detect the tilt of the board and the LED display would be used to display the level. We also were tasked with using the PICO SDK and CMake to build the project. After countless hours of debugging and troubleshooting, we were able to get the accelerometer to communicate with the PICO and display the level on the LED display.

Design

The design of the project centered around the control logic of determinig if the accelerometer was tilted and in which direction. The accelerometer was connected to the PICO via I2C. The PICO would then read the data from the accelerometer and determine the tilt of the board. The PICO would then use the tilt data to control the LED display. The LED display would then display the level of the board. The design was implemented using the PICO SDK and CMake. The PICO SDK was used to interface with the accelerometer and the LED display. CMake was used to build the project.

Implementation

The following code is the implementation of the design. The code is written in C++ and uses the PICO SDK to interface with the accelerometer and the LED display. The code is broken down into three main sections: the main function, the accelerometer interface, and the LED display interface. The main function is the entry point of the program. The accelerometer interface is used to read the tilt data from the accelerometer. The LED display interface is used to control the LED display.

```
  #include "../libraries/lis3dh.h"
#include "../libraries/neomatrix.h"
#include "pico/stdlib.h"

int main() {
  stdio_init_all();
  // Singleton for the accelerometer
  printf("Setting up accelerometer\n");
  lis3dh accel;
  // initialize the accelerometer
  sleep_ms(5000);
  printf("Initializing accelerometer\n");
  accel.init();

  NeoMatrix neo(8, 8);
  printf("Initializing NeoMatrix\n");
  neo.init();
```

```
18
19    while (1) {
20      // Update the accelerometer values
21      neo.clear();
22      accel.update();
23      printf("X: %.3fg\n", accel.get_x());
24      printf("Y: %.3fg\n", accel.get_y());
25      printf("Z: %.3fg\n", accel.get_z());
26      // Set the pixel color based on the acceleration
27      if( (accel.get_x() <= 0.1 && accel.get_x() >= -0.1) && (accel.get_y() <= 0.1
            && accel.get_y() >= -0.1) && (accel.get_z() <= 1.1 && accel.get_z() >= 0.9)
            ){
28        // Set all pixels to red
29        for (int i = 0; i < 8; i++) {
30          for (int j = 0; j < 8; j++) {
31            neo.set_pixel(i, j, 0xFF0000);
32          }
33        }
34      } else {
35        for (int i = 0; i < 8; i++) {
36          for (int j = 0; j < 8; j++) {
37            neo.set_pixel(i, j, 0x00FF00);
38          }
39        }
40      }
41      sleep_ms(25);
42      neo.write();
43    }
44    return 0;
45  }
```

Listing 1: Main Function

Conclusion

The project was a success. We were able to communicate with the accelerometer and the LED display. We were able to read the tilt data from the accelerometer and display the level on the LED display. The project was a success, but it was not without its difficulties. The PICO SDK and CMake were difficult to work with. The documentation was not very helpful and the error messages were not very descriptive. We spent a lot of time debugging and troubleshooting the project. We also had to work with the hardware, which was difficult to work with. The accelerometer and the LED display were difficult to interface with. We had to spend a lot of time reading the datasheets and the documentation to figure out how to interface with the hardware. Overall, the project was a success, but it was not without its difficulties.

One of the big challenges I overcame was working with the PICO SDK and CMake. Many times examples and guides were different and did not explain the benefits or drawbacks of using one commands or another. I had to spend a lot of time reading the documentation and experimenting with different commands to figure out how to build the project. I also had to spend a lot of time reading the datasheets and the documentation to figure out how to interface with the hardware. The hardware was difficult to work with and the documentation was not very helpful.