CECS 447 Spring 2025

Final Project (Group 9)

By

Nicolas de Greef, Cesar Hernandez,

James Sanchez, Raul Vargas

May 7, 2025

Goal:

The goal for this project is to practice I^2C communication and use it to familiarize ourselves with several components that are interfaced with I^2C such as the TCS34725 color sensor module, an MPU6050 IMU, and a 16x2 LCD.

Introduction:

The main function of this project is to use the MPU6050 IMU to move the position of a servo motor that has an RGB paper attached. The position of the servo motor will change what the TCS34725 color sensor module reads. At any time the user can press switch 1 to start the color detection process and then switch 2 to display that color on the onboard RGB light and spell it out on the LCD module and terminal.

Operation:

This project has many different modules that are expected to work in tandem, so we test each one individually in the main to make sure they all work. The first test checks that our timer is accurate by turning the onboard LED on and off based on the wide timer. Next, we ensure that UART is working by calling the sprintf() function so the user can type in the console. After that I^2C is initialized and checked that it can receive information. Finally, once those are all functioning as expected, we can check the peripheral components one by one, making sure the IMU acts normally, the color sensor is detecting colors accurately, the LCD is properly receiving parallel information, and the servo goes to the desired angles. If all of these tests pass, we can use the part 2 mode and use the system wholly as intended.

Theory:

The theory of the final project is backed by I^2C but we needed to be comfortable with several other new components too to make the whole system function. I2C is a communication scheme this project is designed for us to practice. The main feature we are using it for is that we can have multiple slave devices send/receive data on the same bus. Theoretically, we could have multiple masters running on the same bus too but we focused on one master to communicate with our IMU, LCD, and TCS color sensor. Conveniently, I^2C uses half as many wires as SSI/SPI, for example, which is convenient but adds a level of complexity to configuring our system. Each slave and master must have an address on the bus and must include addresses when sending/ receiving messages so all components can identify which signals are meant for them and who sent the signal. Our schematic reflects how much our system is simplified, using only two wires to communicate with 3 different components, which would have used far more pins and wires if we didn't use I^2C.

As for the actual components, the servo motor is the only one that did not use I^2C. Because servo motors only need a ground, power, and signal wires, and the signal wire interprets duty cycles, the only onboard module we had to configure was the PWM generator. Once PWM was configured it was easy to set variables corresponding the correct angle with the correct duty cycle and implement our code straightforwardly.

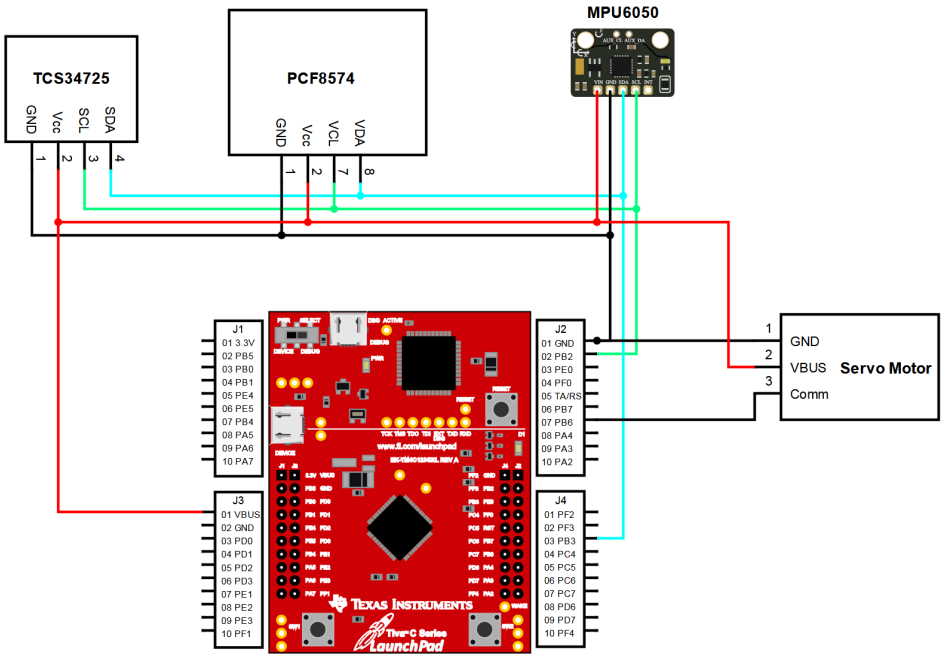
The first component we implemented on the I²C bus was the TCS34725 color sensor. Interfacing with the board is as I have described, only needing a data and a clock pin. This module works by detecting what light is reflected off of nearby objects using photo detectors. These photo detectors are specified for red green and blue and can be used together to interpret which of those colors the sensor is receiving the most. Once the sensor has determined which color is most predominant near it, it puts that data in a package and returns it to the TM4C for us to use.

Next, we integrated the MPU6050. Like the color sensor, the MPU communicates on the I²C bus but measures acceleration and movement rather than color. The MPU has 2 main features, the gyroscope and the accelerometer. The gyroscope measures the amount the module has moved angularly, while the accelerometer checks for changes in position. Together, it is able to give a pretty good estimation of its position in 3d space and saves it as variables of gyro movement and accelerometer movement. Once our microcontroller received those variables on the bus, we interpreted them and used a formula to get the actual degrees of change in the x and y planes to be used later.

Finally, we made the LCD work, which was quite easy as we already did this in a different class. The only real change is the PCF8574 I²C GPIO expansion header, which we used to save pins on the TM4C. All the functions were mostly the same, we just had to package them differently to be sent on the bus.

We've already done this in other projects and are comfortable with it but it should be mentioned we used UART so the micro controller can communicate with the terminal on our computer and send commands.

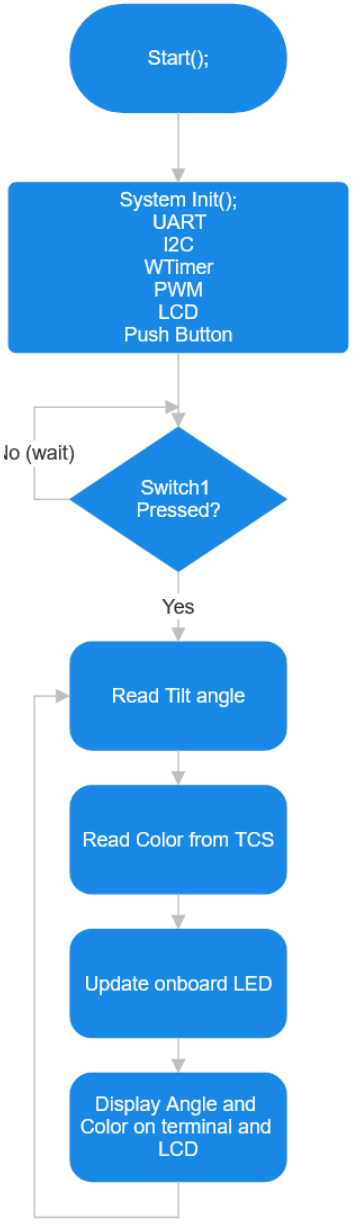
Hardware Design:



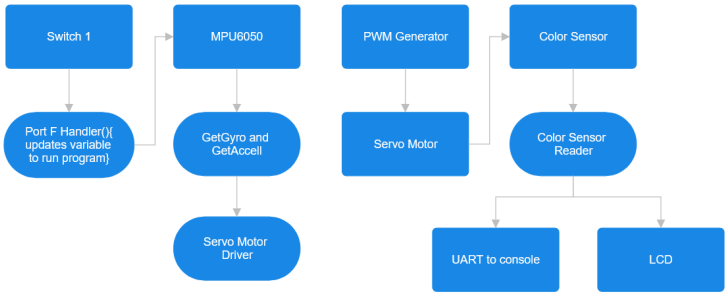
Software Design:

Software design for this project was relatively straightforward, making sure each component individually works was the challenging part. This flowchart explains the main system, just looping through checking sensors and updating outputs after.

With that in mind, each software module is very basic too. Initializing all components correctly was a challenge, but once they turned on they all had predefined functions that were very intuitive.



Data Flow Graph:



Video Demo

https://drive.google.com/file/d/12HdKrkgjvu_mJERjpZrSGPd9QaGN5t97/view?usp=sharing

Conclusion

In conclusion, we were able to successfully integrate multiple components using the I^2C module on our TM4C microcontroller. We individually validated each component to ensure it would work as one cohesive system. This project helped us build on skills we were already comfortable with, such as UART, LCDs, servo motors, and timers, and develop skills with new components like IMUs and color sensors. Most importantly though, we learned how to balance many external components functioning simultaneously and how to manage resources and signals. Overall, this final project was a good capstone of all the different skills we've mastered in the embedded systems pathway.