Automated Testing Station for Sensing Applications

Design Document

Team Number sdmay23-41
Client/Adviser: Moneim Ismail
Team Members/Roles:
Thomas McCoy: Team Organizer/Software Developer
Garth Anderson: LED PCBs Developer
Malvin Lim: Gas Regulation PCB Developer
Matthew Rief: CAD Design Developer
Team Email: sdmay23-41@iastate.edu
Team Website: https://sdmay23-41.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

IEEE: Code of Ethics and Software Engineering Standards

ASME: Boiler and Pressure Vessel Code

IPC-2221 Standard in Circuit Board Design

Summary of Requirements

Functional requirements:

1. The device must be able to collect I-V, which is the reflected light from the device under
testing (DUT).
2. The device can shine LEDs with different wavelengths desired by the user from the top
and bottom of the DUT.
3. The device could provide a different testing environment for DUT by changing the type of
air in between sensors and the DUT.

Resource requirements:

1. PCBs
2. Servos
3. Probes for the DUT
4. Box that can hold a vacuum and block out light.
5. Gas regulators/valves & tanks

UI requirements:

1. Provide tables of processed data
2. Easy to navigate
3. Gives option to view raw data

Applicable Courses from Iowa State University Curriculum

EE 333 - Electronics Systems Designs.

New Skills/Knowledge acquired that was not taught in courses

LabView programming.

Table of Contents

1 Team

1.1 Team Members

Thomas McCoy

Garth Anderson

Malvin Lim

Matthew Rief

1.2 Required Skill Sets for Your Project

PCB design principles

CAD software

Gas regulation principles

Software design principles.

1.3 Skill Sets covered by the Team

Software design principles - Thomas McCoy

PCB design principles - Garth Anderson and Malvin Lim

CAD software - Matthew Rief

Gas regulation principles - Malvin Lim

1.4 Project Management Style Adopted by the team

Waterfall design approach

1.5 Initial Project Management Roles

Thomas McCoy - Team Organizer / Software Development

Garth Anderson - PCB Research & Design

Malvin Lim - PCB Research & Design

Matthew Rief - CAD Designs

2 Introduction

2.1 Problem Statement

Our client needs a more efficient and automated way to test different photosensitive devices and how they respond to different wavelengths of light through a medium of different gasses. Currently, when testing these devices' capabilities, the process is manual, making it arduous, expensive, and time-consuming. We will solve this problem by automating the testing process and collection/interpretation of data.

2.2 Intended Users and Uses

User 1: Perovskite solar cells researcher

1. Key characteristics - Research on solar cells currently available in the market and improve the efficiency of the solar cell.
2. Needs related to the project - A more efficient way of testing the solar cell or solar panel.
3. How they might use or benefit from the product you create - They could use this project to do a test on the solar panel without any knowledge of operating the automated testing device.

User 2: Photodetector manufacturer

1. Key characteristics - Produces mass quantities of precise photodetectors that must adhere to specifications
2. Needs related to the project - Photodetectors from each manufacturing run should be tested to ensure quality of the product is consistent and meets specifications.
3. How they might use or benefit from the product you create - It would allow for the manufacturing process to get faster feedback on the quality of their process if they can quickly test units, and make adjustments to the process if needed.

User 3: Students

1. Key characteristics - Students attending the university that are studying photosensitive devices.
2. Needs related to the project - Needs the machine to be intuitive and flexible for testing devices.
3. How they might use or benefit from the product you create - Students can experiment with different devices and learn about how testing these types of devices works.

2.3 Requirements & Constraints

Functional requirements:

1. The device must be able to collect I-V, which is the reflected light from the device under testing (DUT).
2. The device can shine LEDs with different wavelengths desired by the user from the top and bottom of the DUT.
3. The device could provide a different testing environment for DUT by changing the type of air in between sensors and the DUT.

Resource requirements:

1. PCBs

2. Servos
3. Probes for the DUT
4. Box that can hold a vacuum and block out light.
5. Gas regulators/valves & tanks

UI requirements:

1. Provide tables of processed data
2. Easy to navigate
3. Gives option to view raw data

2.4 Engineering Standards

IEEE: Code of Ethics and Software Engineering Standards

ASME: Boiler and Pressure Vessel Code

IPC-2221 Standard in Circuit Board Design

3. Design

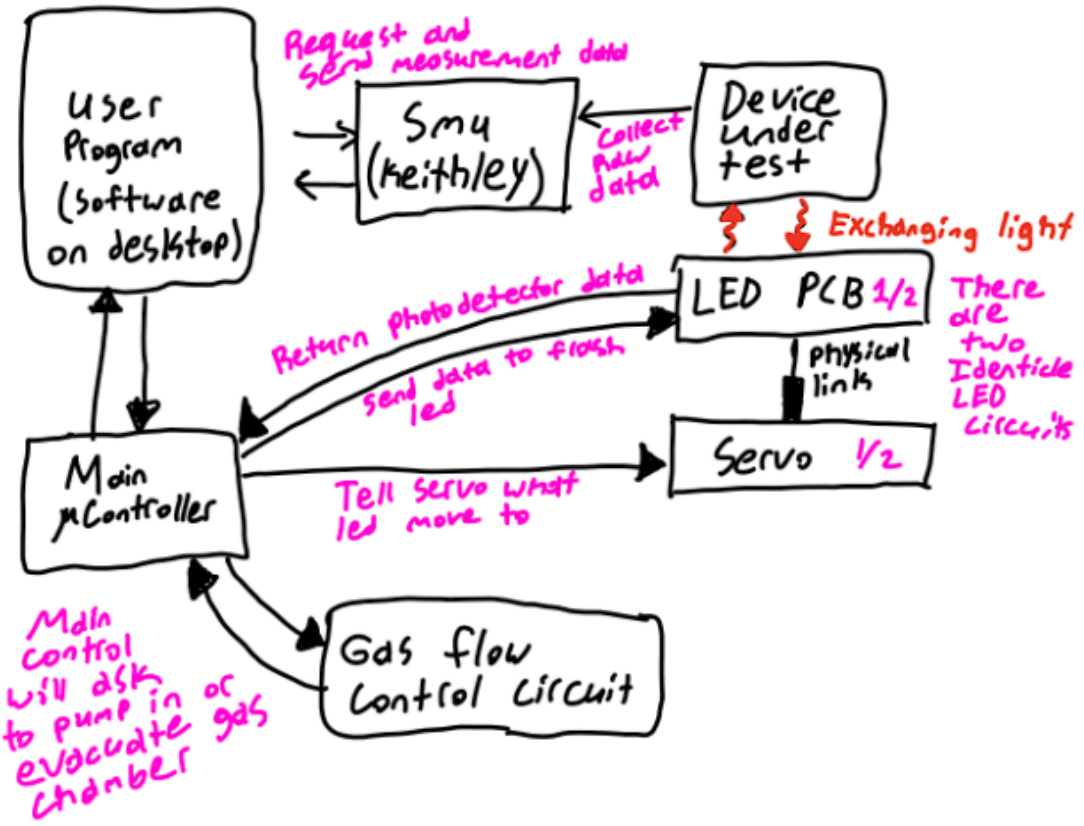
3.1 Initial Proposed Design

3.1.1 Overview

Our Automated Testing Station is composed of a few subsystems. Our device includes the LED PCBs, the gas regulation system, the DUT(device under testing) probe system, and the software interface that interprets the data and controls certain components. The LED PCBs are composed of two identical PCBs mounted to servos. The PCBs have three different LEDs of different wavelengths that can be pulsed and a photosensor. The servo allows the PCBs to be rotated by the servos to allow the different LEDs to be used for testing. The gas regulation system controls the gas flow rate into the testing environment and also allows for a vacuum within the testing environment. This system contributes to the ability to test different devices in the presence of different gasses. The DUT probe system allows for collecting raw data from the DUT, which the software interface can then interpret. Lastly, the software interface communicates with each component, allowing for control of certain features. It also collects data from the DUT and interprets and graphs that data.

3.1.2 Detailed Design and Visual(s)

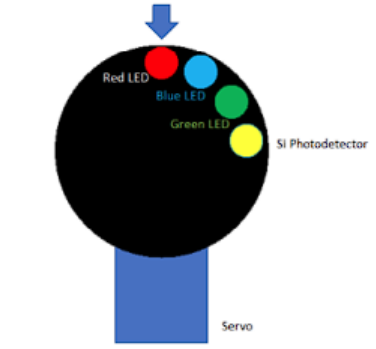
Systems level design

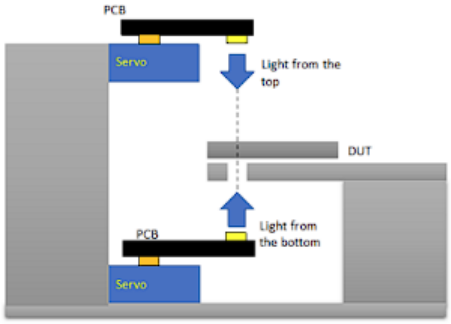


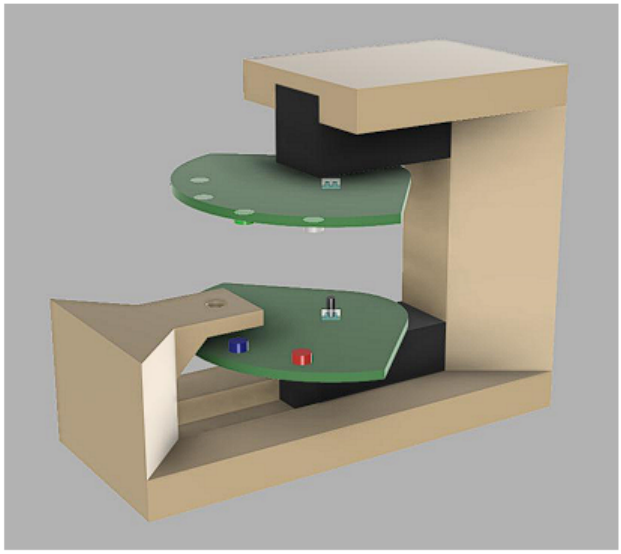
Above is a systems level diagram of our proposed solution. The user will load the device under test (DUT) into the machine and seal it up. From there they will begin interacting with the user interface software running on a desktop computer. The program will be a lab view interface that allows the user to graph and analyze data as well as control the machine to take measurements. When instructed to make a measurement by the user, the desktop software will communicate with a main microcontroller that will send directions to subcomponents to begin making measurements. To make a measurement the gas control circuit will be instructed to either vacuum out the testing chamber or fill the chamber with air or another provided gas (note the machine will sit inside an airtight chamber). From there it will instruct one of the two servos (top or bottom) to move the proper LED or Photodetector into position under/over the DUT. It will then instruct the LED PCB to flash the LED to stimulate the DUT with light. At this point the user interface will also need to talk to a source measurement unit (SMU) that will periodically take electrical current vs voltage measurements of the DUT, which will be solar cells for most types of test. For a specific type of test the DUT will be an LED or some other device that will generate light itself, the intensity of which can be measured by a photodetector on one of the PCB boards. The main controller will then send data from the

photodetector back to the program. Finally the program will automatically perform analysis and graph relevant information for the user.

Physical design







Above are initial design concepts for the physical assembly of the core of the machine. As you can see the two PCBs will be mounted above and below the DUT which will be sitting on a stage in the middle of the device. The main structure will be 3-d printed, and will include mounting holes to secure the servomotors. The LED flasher PCB will sit on top of these servos. (Not Pictured) This core assembly will be placed inside of an airtight, blacked out chamber that will also house PCB's for the the gas control circuitry and main servo controller. Additionally there will be many wires connecting for power and communication between the various components. As well as wires leading through airtight junctions of the chamber for connecting the SMU multimeter leads to the DUT, and connecting the main microcontroller back to the desktop running the user interface.

3.1.3 Functionality

The device we are building is called an Automated Testing Station, and the main purpose of this device is to get the data and information required when testing a solar panel. The operation of the device would be very simple. The user will place the device under testing (DUI) in a bracket and use a computer to control the testing environment. To control the testing environment, the user can choose the light frequency and the type of gas going into the testing device. By choosing that, the user can understand how the DUI would react in different environments. After choosing the light frequency, the motor responsible for the light frequency will rotate to the angle so that the chosen LED will be placed directly above and under the DUI. Then, after the type of gas is selected by the user, the device will input the gas into the device while the flow rate of the gas device will appear on the LED screen that will be on the device. Then, sensors placed onto the PCB will collect data and send data to the Microcontroller, which will then show complete data in the computer.

3.2 Design Evolution

Changes for Mass Flow Controller (MFC) PCB.

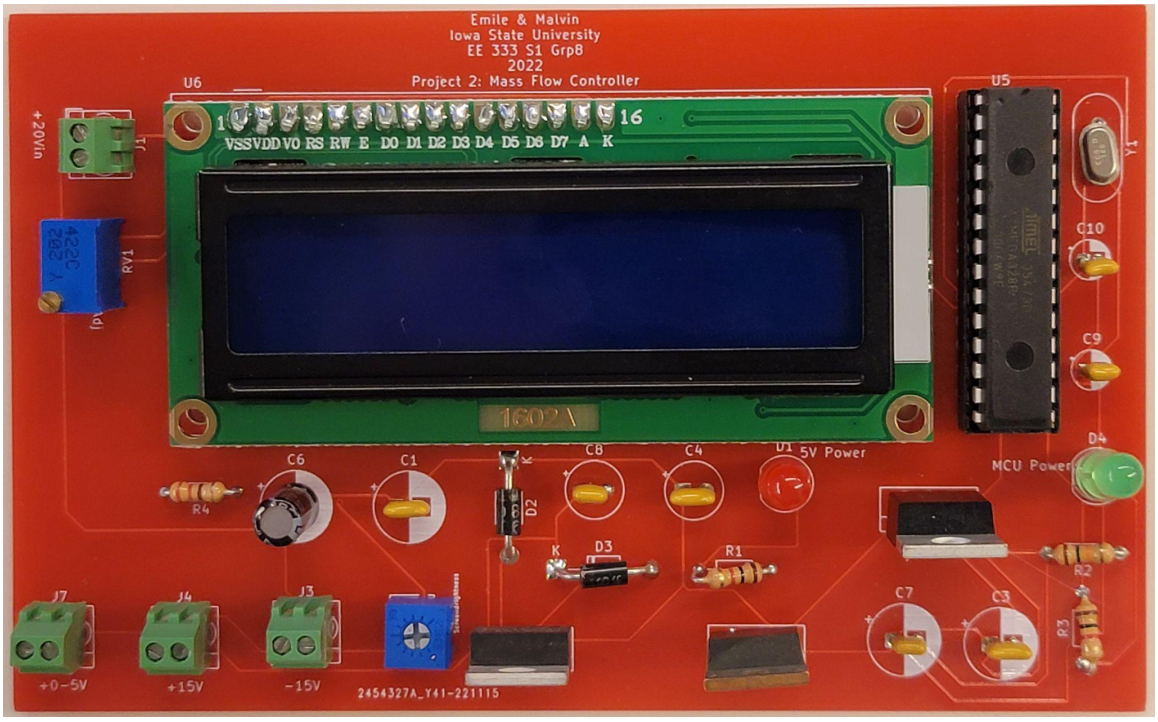


Figure: MFC PCB from the last prototype

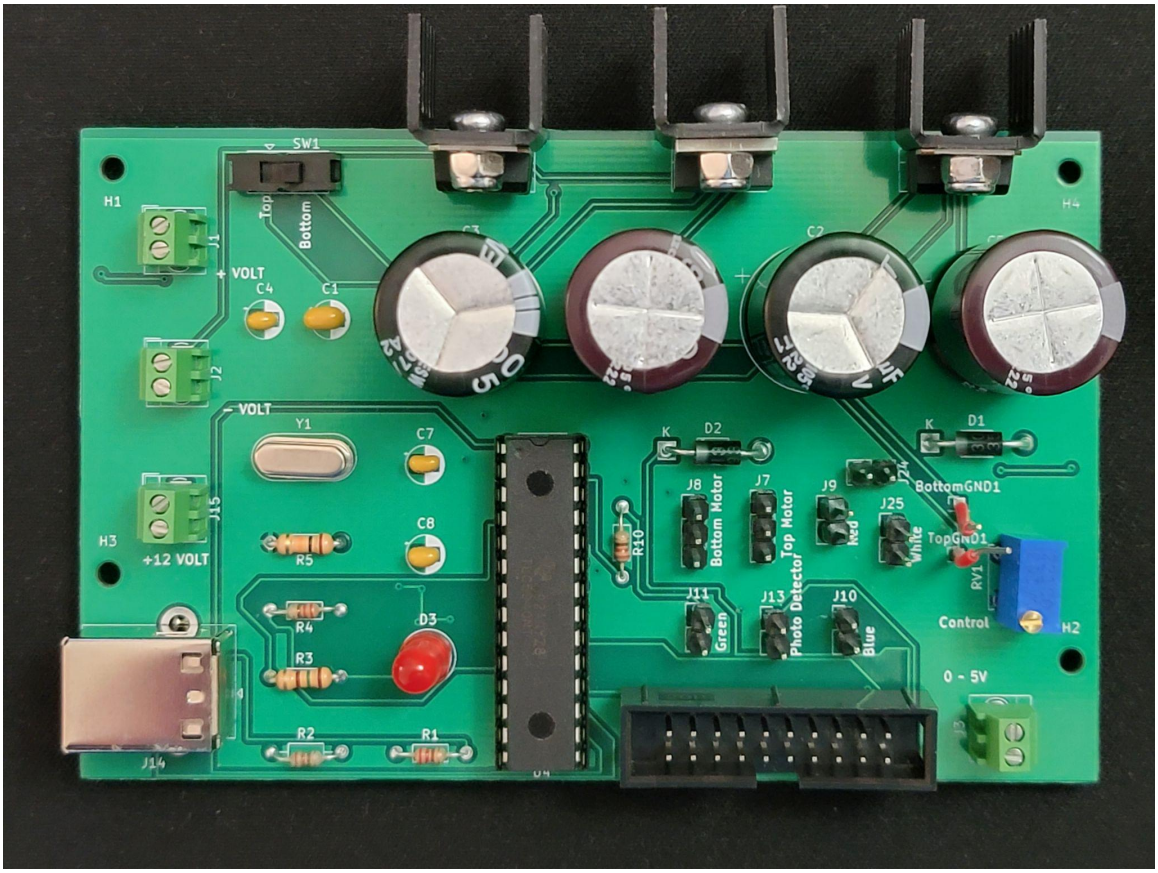


Figure: MFC PCB for the current prototype

This semester, I have made a few changes to the Mass Flow Controller PCB I designed last semester. The first big change I have made is replacing the LCD screen with a USB B port, which could save some space on the PCB to make it smaller and more cost-efficient to produce this product. Replacing the LCD screen with a USB B port, we can connect the PCB to a computer to show all the information the Microcontroller unit collected.

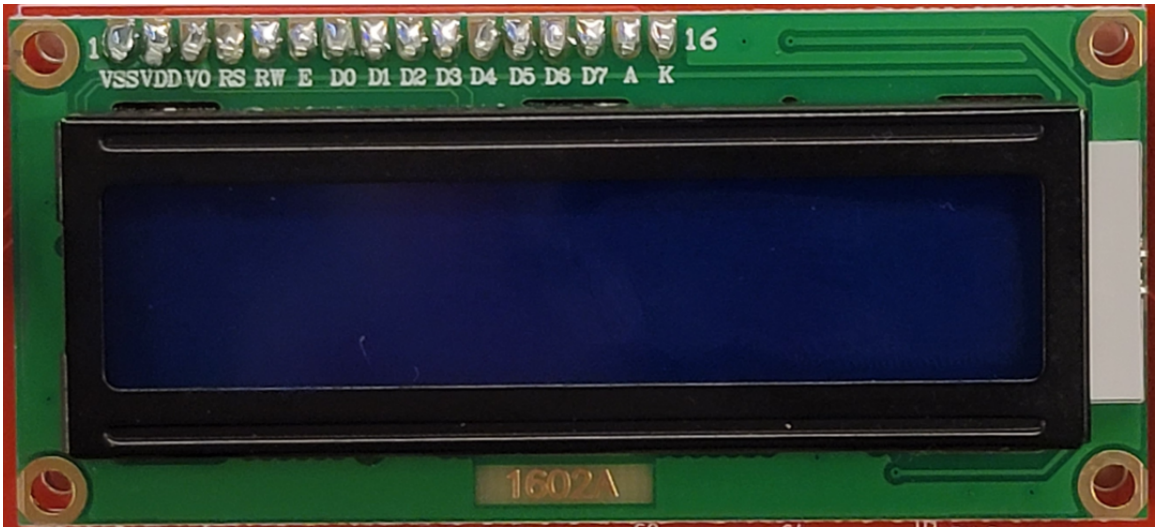


Figure: LCD screen used for last PCB

Figure: USB B port we are using on the MFC

Then, we changed the input from one input to three separate inputs. Doing so gives us a more stable result than only one input. Also, we created three different grounds for this PCB, which are for +15V, -15V, and 5V regulators. By creating three different grounds, we can avoid all of the voltage to sum up to one, which could cause trouble toward the result.

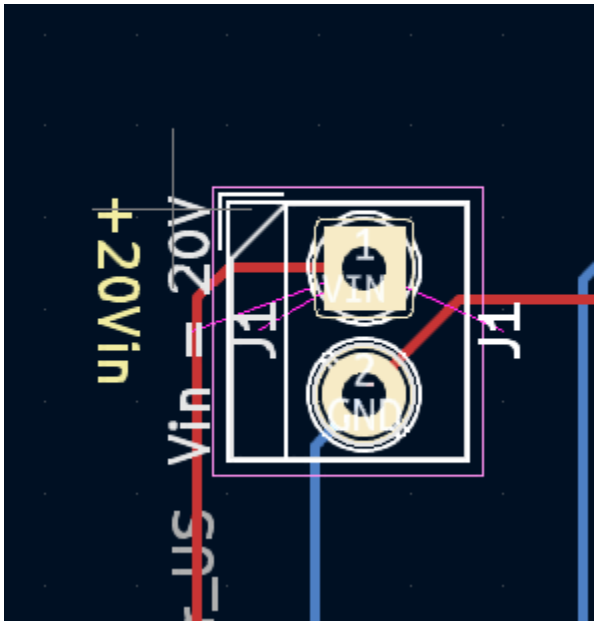


Figure: +20V input from the last prototype

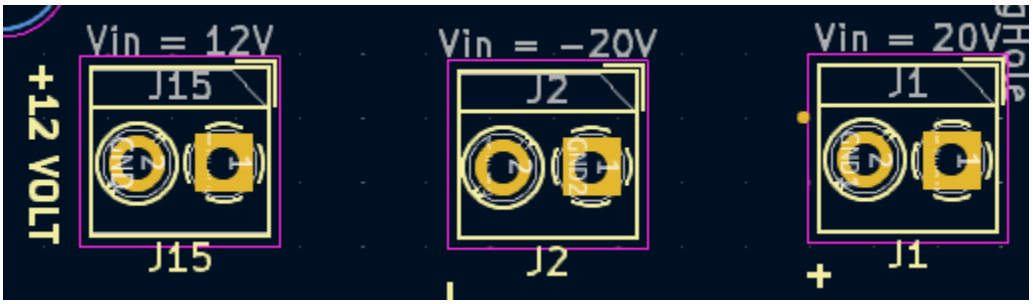


Figure: +20, -20, and 12V input in the current prototype

Then, we implemented a significant change by replacing the two-pin power jacks with 20-pin jacks to power up and control the Mass Flow Controller device. This upgrade included multiple analog pins, three grounds, and two power inputs designed for +15V and -15V supplies. Using a 20 pins jack also matches the client's needs, as he preferred to use a 20 pins jack on this project.

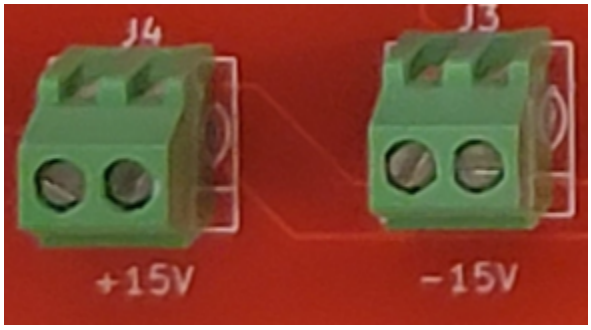


Figure: 2 pins jacks

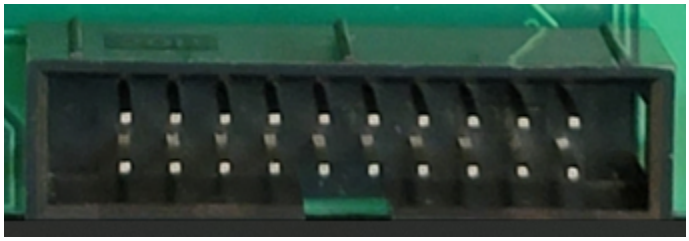


Figure: 20-pin jack.

Merging part of LED and Motor PCB into the MFC PCB.

This semester, we also merged a part of LED and Motor PCB into the MFC PCB as we could save some cost of producing it and make it easier to operate by the user. By doing so, we also added some complexity to this project, as there are a lot of things that will be added as well as removed. The most significant reason for doing this is to use only one Atmega chip which acts as a Microcontroller unit for our project. Now, we only require one working Atmega chip to control both PCBs, which could control the mass flow of gasses, rotate motors, and shine LEDs.

Two LED and Motor PCBs.

We also added another LED and Motor PCB in this project, which will work simultaneously. By doing so, we could collect better results from the Device Under Testing (DUI) as there will be two ways to shine and collect from top and bottom. Since 2 PCBs need to be controlled and there are limited pins on the Atmega chip, we decided to use a switch to switch the connection between controlling the top and bottom PCBs.

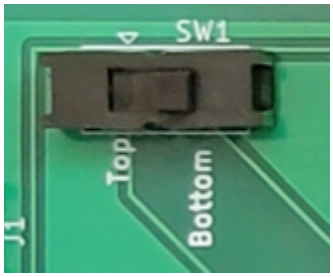


Figure: Slide switch used in this PCB.

Then, we will need to use female-to-female connectors wires to connect three PCBs, which leads us to add some connector pins to the MFC PCB. Each LED light on the PCB will require one pin; therefore, we will need eight pins with two grounds for LED as red, green, blue, and white require powers.

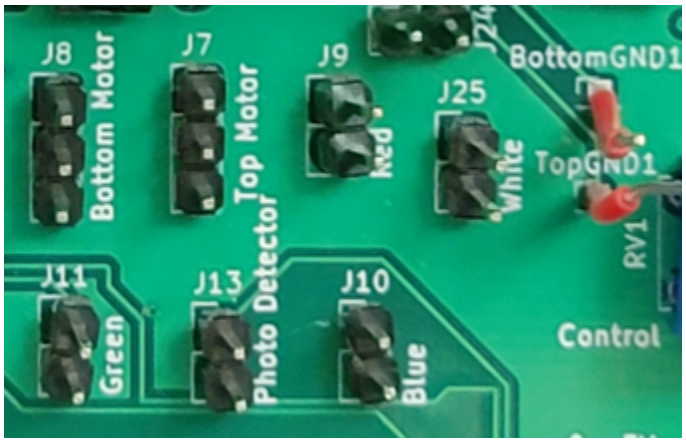


Figure: Connection pins on MFC.

Proto shield.
We also designed a PCB, which will work as a backup plan if the main board does not work as we want. It's a proto shield that will be stacked onto an Arduino Uno. The peripheral extensions required in this proto shield are +15 and -15V regulators, as they could not produce voltage greater than 5V. After adding the positive and negative voltage regulators, it could act as the Mass Flow Controller.

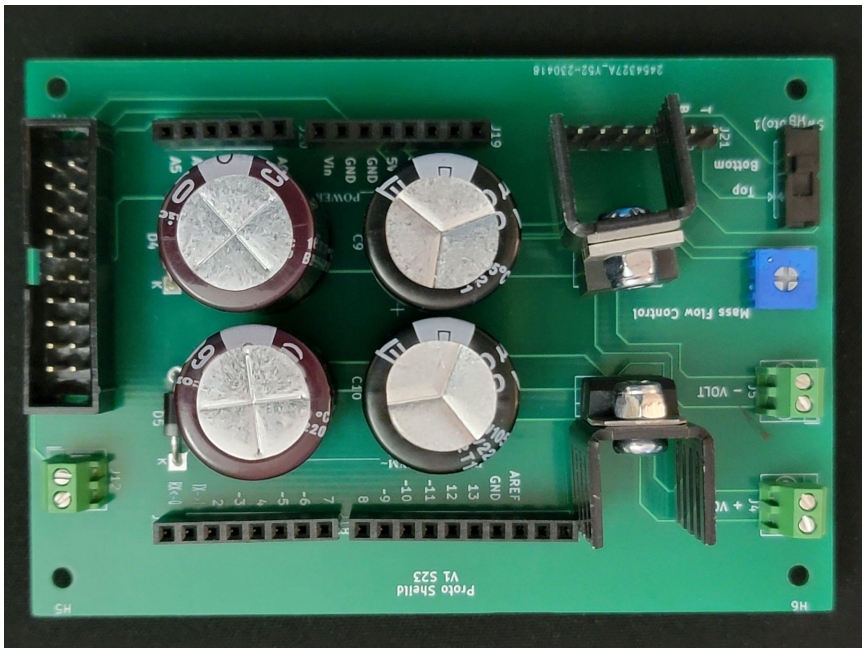


Figure: Proto shield for MFC.

4 Testing

4.1 Software

4.1.1 Unit Testing

The software controls can be broken down into three distinct components:

LED control, Servo control, and Measurements.

These different "units" are tested by breadboards, and the keithley

4.1.2 Interface Testing

The combined interface is tested via the front-panel provided by LabView along with the use of breadboards for the testing of the LED and Servo control at runtime.

4.1.3 Results

Unit testing showed that individual components functioned, leds turned on, and could blink at different frequencies. Servo's could rotate a specified amount of degrees, and the Measurement data appeared to be correct via the Keithley data.

For the interface testing, combining all the components show that the front panel is allowing for complete control of LEDs, Servos, and Measurement data at runtime as specified by our project requirements.

4.2 Hardware

4.2.1 Unit Testing

Components testing:

After getting all the components required for this project, we have to test each of the components we will be using to ensure that they will work without issue after we solder them onto the PCB. To test all of the components, we have to build each circuit onto a breadboard and use devices available in the lab. Two main measuring tools that we used were a power generator and a digital multimeter.

First, we tested linear regulators, capacitors, and diodes by building +5, +15, and -15V regulators on a breadboard. By doing so, we have done testing for LM7915, LM7815, LM7805, and 1000uF & 100uF capacitors. We also found that we need to change some footprints on the PCB as they do not look the same as the one we thought.

Then, we tested the Atmega chip by replacing it with the Atmega chip on an Arduino Uno. Every Arduino Uno comes with an Atmega chip, as it acts as a microcontroller unit. To check if an Atmega chip is working, we first need to see if the Arduino Uno is powering up. Then, we could run a simple code like pulsing a LED light with the Atmega chip. If everything works fine, then the Atmega chip is good.

4.2.2 Interface Testing

The layout of the PCB has been printed out on paper to check the footprint of each of the components. Then, we place components onto the paper to see if they fit onto the PCB.

4.2.3 Results

From the testing that we have done, we can confirm that every component is working and ready to be soldered onto the PCB. Then, we update some of the footprints on our PCB to match the components that we have.

5 Implementation

5.1 Software

The software solution provides an interface which controls the different hardware components. It also allows for the measurement of data. The software interface is implemented via LabView. The program provides a "front-panel" which acts as an interface, and a block diagram which comprises

the code. The interface allows for servo configuration, LED selection and configuration, and measurement configuration and results.

5.2 Hardware

The hardware for this project consists of three PCBs and a backup proto-shield. Two PCBs are mounted on motors inside of the device, one above the DUT and the other below. These PCBs flash LEDs at the DUT and rotate to align a photodetector with the DUT to measure light emitted. The photodetectors send analog signals to the external PCB, which is the controller board for the entire hardware system. The external PCB includes an Atmega microcontroller which uses PWM signals to drive the LEDs on the internal PCBs, as well as to control the motors. The external PCB also controls the mass flow controller and includes a mechanism for switching between the top and bottom internal PCBs. The backup proto-shield is a PCB which can be connected between our device and an Arduino Uno board in order to control our device if our controller PCB fails.

6 Appendices

6.1 Appendix 1: Operational Manual

Setting up the Software:

The software is entirely independent from the hardware components of this project, while it is used to control different hardware components, it does not rely on any hardware components. Because of this, a setup of the software without hardware components is possible.

Install LabView:

LabView is a graphical programming language which allows for a very streamlined development of interfaces for hardware components. It is required in order to run/test the software interface.

Download LabView program:

The program is of the file extension .vi which is shared with all other LabView programs. Once loaded into LabView, you can open the .vi and view the front-panel and the block diagram.

Configuration:

The interface allows the user to configure many different settings before or during runtime.

These different configuration options are broken down into different sections:

1. VISA Resources
2. Servo Options
3. LED Duty Cycles
4. LED Options
5. Measurement Configuration

These sections allow for the complete control of the different features provided by our project.

VISA Resources:

Allows the selection of the VIA resources which primarily includes the Arduino Uno, and the Keithley devices. This must be set before program runtime.

Servo Options:

This section allows the user to select top/bottom for which servo they want to send commands to. And a selection between LEDs/photodetector for the degrees of rotation to send the selected servo. This is to be used at runtime.

LED Duty Cycles:

This section gives complete control of LED brightness and color control. The different LED channels are broken into 4 different sliders. These are designed to be used at runtime.

LED Options:

The interface also allows the user to select if they want the LEDs to be either solid or blinking, and if blinking the frequency at which they should blink. This is limited between 0-60Hz, but is changeable via the block diagram. These settings can be changed at runtime.

Measurement Configuration:

This section allows the user to set several important settings. This includes the timeout length for measurements, the amount of time between subsequent measurements (could be a number of seconds, or even a number of days). It allows the user to select the source mode, the minimum and maximum amplitude, and the number of points displayed. It also shows what the default settings are. Above these settings includes a graph which will display measurement data. These settings can be changed at runtime, but it is recommended that they are set before starting the program.

Testing

Each component can be tested individually provided access to an Arduino Uno for non measurement components, and a Keithley 2400 for the measurement component.

Non-measurement components

This includes the Servo and LED configurations. To test these components, attach an Arduino Uno via the VISA resource to a computer running the LabView program. For the LED testing, utilizing a breadboard, attach wires from digital channels 3, 5, 6, and 11 to Red, Green, Blue, and White LEDs respectively. Once done, you can either attach the Keithley or suppress the VISA warning for not having it connected, and run the program. Moving the LED sliders should visually brighten/dim the respective LED. For Servo testing. attach wires from digital channels 11 and 9. Again follow a similar process for the LEDs, start the program, and utilize the front-panel buttons and see if the servos function according to specification.

Measurement component

This includes the keithley measurement to test this component, you will first need something to measure that has a predictable result. Once this is accomplished. Attach the VISA resource for the keithley and connect the probes prior to running the program. Next set measurement configurations via the front-panel, and finally run the program. Much like with the non-measurement components, if you don't provide a VISA resource for the Arduino Uno, it will give an error, which must be suppressed in order to run. The program should then output measurement data to a graph which can be compared with the expected results.

6.2 Appendix 2: Alternate Designs

One of the alternate designs discussed was to only use one servo instead of two. In this case, the two LED PCBs would be attached by a rod at the center, and rotate together. This idea was scrapped due to time constraints with the project; several designs were already implemented with the two servo design.

Another alternate design decision that had to be made was to move the potentiometer from the software to the hardware, the specifications of the Arduino Uno does not include the ability for an analog write, therefore making it extremely difficult to do via the Arduino and software.

The initial plan was to have a separate microcontroller for each of the PCBs mounted on motors. This plan changed to using one microcontroller on an external PCB with jumper wires connecting to each of the PCBs on motors. In order to implement this system without running out of digital signals to drive LEDs, we use one signal for each respective color (RGB) on both PCBs simultaneously. Since we only want one board to be shining LEDs at a time, we implemented a switch which toggles a common ground between the two boards, effectively activating one board by connecting ground, while deactivating the other board by disconnecting ground. This same system was implemented for the photodetector circuits.