COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

RoboReform

Alex Morch
Angelina Mauleon
Daniel Kenehan
Katie Lam

Revised December 12, 2025

CSCI 370 Fall 2025

With advisor Mr. Scott Jenson

With client Professor Iris Bahar

Table 1: Revision history

Revision	Date	Comments
New	8/29/2025	Added introduction, requirements sections, risks, definition of done, and team profile.
Rev – 2	9/19/2025	Added System Architecture, Software Test and Quality, and Ethical Considerations. Updated Introduction and Functional Requirements.
Rev – 3	12/11/2025	Revised sections that need updating, finalized for the final report submission.
Rev – 4	12/12/2025	Finishing Touches

Table of Contents

I. Introduction

Engineering isn't easy. Whether in school or in the real world, engineers feel every emotion under the sun, from the frustrations of a project not working, to the joys of a project finally working. Instead of pushing these feelings down and isolating them, we aim to embrace them and bring a community together by showing you aren't alone in how you feel, through art. We want to show that engineering and art can intersect through The Emotional Flowerbed.

The Emotional Flowerbed was originally created by a Capstone Senior Design team in the 2024-2025 school year, and then passed onto us due to it feeling incomplete. While the previous team did create a working and functional art piece, our client, Iris Bahar, the department head and professor of computer science at Colorado School of Mines, felt that more could be done with the project. Unfortunately, in the process of passing the project down, the flowerbed underwent some damage. Our team was tasked with repairing The Emotional Flowerbed to its previous working state, and then adding/updating features and code so that the flowerbed is more interactive and is easier to add to in the future. Our client's ultimate goal for this project is to promote empathy and display users' emotion through interactions with the flowerbed.

We aim for Mines students, professors, and guests to interact with the flowerbed and use this software. This project will then be maintained by our client and any future group who picks up this project after us.

II. Functional Requirements

For The Emotional Flowerbed to work, we have the following core functional requirements that our flowerbed must be able to do in order to meet the needs of our client, our team, and future users.

- The flowerbed should be operational.
- The flowers should sync up with the dial, including turning colors and on/off.
- Music plays when the dial is pressed.
- Music is overlaid as more emotions are added, rather than playing them sequentially
- The large flower opens and closes in response to the dial.
- Functioning website that collects interactions between users and the flowerbed.
- The website is able to turn the data into graphs.
    - Show trends with emotions i.e. if there are more prominent emotions during certain times of the day
- Synchronize the LEDs with the music
- Any additional features are also operational.

III. Non-Functional Requirements

In addition to the previous functional requirements, we also have the following non-functional requirements that we want to achieve for easier use for our client and future teams that pick up this project afterwards.

- Need to have easy access to the wiring inside the table.
- Code is written in Python (for the RaspberryPi) and Arduino C (for the Arduino Uno).
- Make wire management cleaner and more organized.
- Start and maintain a GitHub repository for code history.
- Code is commented, readable, and organized.

IV. Risks

While this may seem like a no risk project, there are a few risks that have come to our team's attention. Starting the project, our team members might not have sufficient knowledge of the previous team's code. This is such an important aspect of the project that if we cannot understand or access the previous team's code, the project cannot be completed sufficiently. Along with this, if the code is not properly contained within a repository, it could get lost. To avoid this, we are taking the time to understand the current code in the flowerbed and how it all connects and then storing the current version of the code in a living GitHub repository where we'll continuously update and store any revisions.

Another significant risk is parts of the physical flowerbed could fail, which would then result in a high cost to replace them. Due to this being a one semester project, we do not have the same funding that the previous team had for being in Capstone Senior Design. If something as large as the main motor of the big flower breaks, that will be an expensive replacement. To manage this risk, we plan to take caution with every action and to not do anything without someone else there in order for quick reactions if something seems to be going wrong.

As learned from the previous team, if the flowerbed is transported then there is a possibility that the flowerbed breaks. Whether the physical container breaks or is damaged, or individual components such as the wiring, the LEDs, the motors or dial, there is a huge risk with transporting the flowerbed. In the end, it would be safer to keep the flowerbed where it currently is.

Finally, there is always a risk that a user incorrectly uses the flowerbed. As engineers, you always have to assume that the user knows nothing and lacks previous experience in order to properly interact with the flowerbed. Incorrect usage can circle back to the previous risks of the flowerbed getting damaged. To mitigate this risk, our team decided to make a poster that will sit at the bed, showing step by step detailed instructions to properly interact with the flowerbed.

V. Definition of Done

In order for The Emotional Flowerbed to be considered done, multiple things need to be complete. First, our team must have done sufficient research for any features we consider adding to have a better idea of what can be done in the time and budget that we have. Not only should these features that we choose to add be fully functional, but all the previous features that the previous team had should also be fully functional. We want to enhance and upgrade the flowerbed, not completely disregard the work that the previous team put into it.

Last but not least, the final flowerbed must meet not only the client's expectations, but also the team's expectations. This includes being satisfied with the result of the website implementation, the improvements to the physical flowerbed, and the music overlays and switches between what our team creates and what the previous team created. In the end, we want to reach the hearts of anyone who interacts with the flowerbed, and the minds of any team who picks up the project after us.

VI. System Architecture

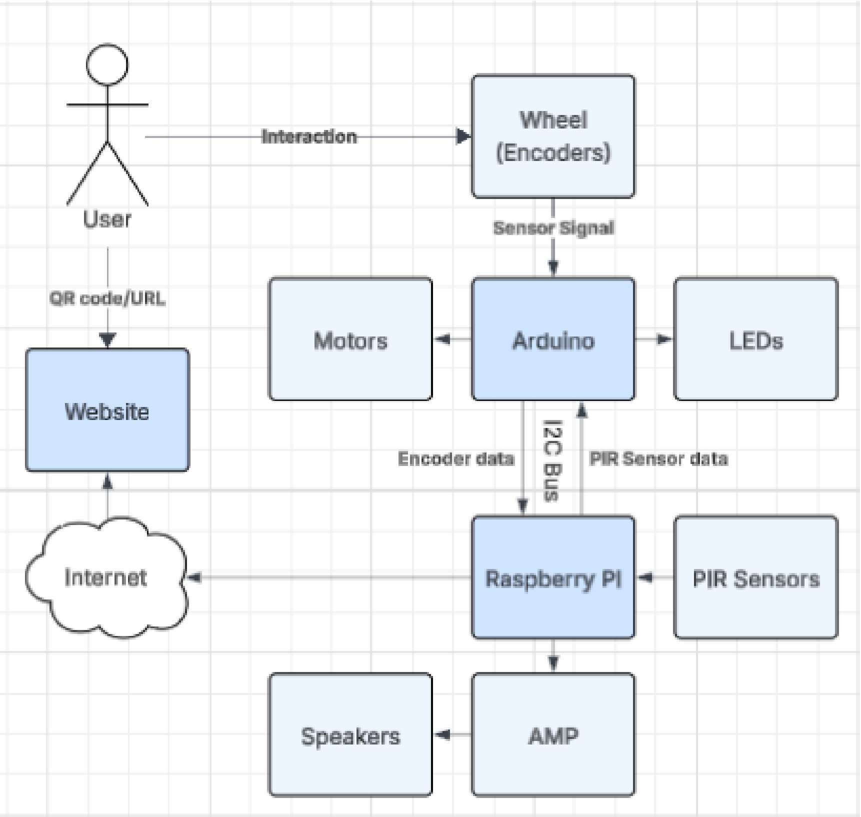


Figure 1: System Diagram

Within the flowerbed, the system contains a few major components: the Arduino and the Raspberry Pi. When a user interacts with the flowerbed, the encoders in the dial will send a signal to the Arduino, letting it know which emotion was dialed. The Arduino then sets the LEDs to certain colors, and controls the motor to either open or close it. It also utilizes an I2C Bus connection to send the data to the Raspberry Pi.

The Raspberry Pi plays music based on the signal from the Arduino. It also takes in data from the many PIR Sensors around the flowerbed. These detect if a person is nearby, and can disable the system if not. Finally, the Raspberry Pi will send data on which emotions have been dialed to our website.

A user can access the website VIA a URL, or a QR code with that URL. The website will display the inputted data in various formats such as a pie chart or a line graph, showing emotional trends across the day, the week, the month, or the year. Along with this data, randomized facts corresponding to the emotions will be displayed, with special facts for holidays.

At the time that we got the flowerbed, there were a few design issues that we had to tackle. One was the inability to edit Raspberry Pi code due to not having the log in information and being unable to consistently connect to wifi. Due to the changing wifi on campus, when the flowerbed moved between buildings from the Capstone Senior Design Showcase to the 2nd floor of CTLM the Raspberry Pi got disconnected from the wifi and unable to connect again without intervention every time. This caused the music synchronization to not work at all. Additionally, when activating the motor for the big flower, the motor would overextend, as in try to open past its limits or close even more than it can. Finally, the PIR sensors were not detecting properly, to the point where we were unsure if it was ever in a working state.

VII. Software Test and Quality

This details the motivation behind how we would go with testing the repairs to the flowerbed to make sure that, if it does not work, there are still plenty of resources available for future students to work on adding their own additions (or fixing any problems with it in the future).

- Code is backed up and stored online
  - Lets people submit additions (or roll back to previous builds if needed) which can then be downloaded onto the Raspberry Pi. This is also where we can do code reviews.
- Making sure Motor doesn't overextend
  - We are updating the previous team's code to make the motor properly collapse the flower/motor so it doesn't break the bed. As well as making sure there is a debug option to manually change the motor.
- layering audio
  - We are attempting to make sure the layered audio plays when different combinations of emotions are selected
- checking LED synchronization matches up
  - We are designing a sequence to make LEDs have a consistent pattern based on each of the different audio files
- Testing website navigation is intuitive
  - We will have prototype websites and examine and iterate based on how different people interact with it
- Making sure project persists long term
  - quick access to Pi and servers helps monitor if there is an internet problem making them go down
- Making flowerbed easy for repairs in the future
  - Code for flowerbed is in Raspberry Pi, and login information is shared to those who wish to make edits

VIII. Project Ethical Considerations

The biggest ACM/IEEE principle that is the most pertinent and also the most in danger of being violated is keeping integrity. With the development of the website, there are ethical considerations that need to be made to keep trust between the users and ourselves. Users' data is being collected in the form of their interaction with the flowerbed as they give input of their emotional state based on the provided options. To apply the Michael Davis legality test, before users are able to interact with the flowerbed, they will be informed that the interaction will be recorded and put onto the website. They will also be told what is being done with the collected data: creating graphs and analysing usage trends. To ensure the data collecting is ethical, users' interactions with the flowerbed will be anonymous.

It is also important to consider the environmental impact the flowerbed is causing. The flowerbed is plugged into an outlet which can consume a lot of power. To be mindful of this, we apply the Michael Davis harm test and verify that the flowerbed is powered off and unplugged when not in use. The flowerbed may also need physical repair done. To reduce waste made during the repairs, we will check between each other that the repairs being made will fix the problem and will not need another fix. We find these to be the best options as it minimizes the usage and waste of resources and energy.

IX. Project Completion Status

Some of our results with updating the flower bed have been in the form of setting up a website to chart user's inputs. The website can be accessed at emotional-flowerbed.web.app. When an emotion is selected from the flower bed, the value is sent to a database which the website then will graph based on how its visitor wants it to be (the website can display emotions throughout the day, week, month, and year).

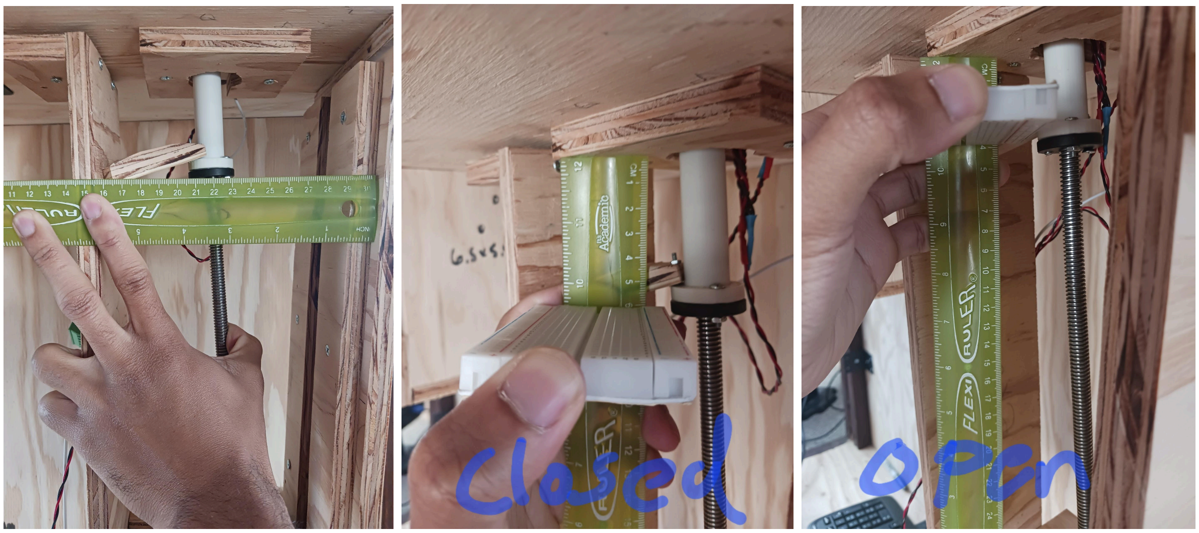
To ensure more safety and maintenance of the flowerbed in the future, we have added some debug options to the flowerbed. These can be accessed by pressing various buttons in the back of the flower bed, and include a debug LED display (which iterates through all of the different LEDs and attempts to turn them on to see if there is a faulty connection), and a motor emergency brake. The plan is to mount the emergency brake button in a way that it would stop the motor from over-extending should it reach that point accidentally. We also did some cable management to make it easier to move the top of the flowerbed without all of the wires snagging or getting loose. This involved gluing the wires of the LEDs for the small flowers to the top of the underside of the flowerbed. We only did these wires since they are less likely to break or need changing. The other wires, like the ones connected to the Arduino, Raspberry Pi, and PIR sensors, were intentionally left free so we can access and adjust them more easily in the future.

Beyond the more apparent and visual changes to the flowerbed, we have also made improvements to the code/documentation of it. Detailing our own additions (like the website) and setting it up so should someone in the future needs to do repairs to the flowerbed, there is a github/drive of documentation that can be referenced.

X. Future Work

Most of the future work that can be done are improvements to a lot of the components mentioned below. Some miscellaneous maintenance things that can be done include better soldering for certain connections, remaking parts of the flowerbed that are redundant. Like using the Raspberry Pi and Arduino, technically you would only need one, but as we are inheriting this from another group they used both. The Raspberry Pi is good for accessing files like the website and music, while the Arduino helps with the physical analog devices like the dial, motor, and lights, for the sake of making changes easier and more intuitive, having the Raspberry Pi control most of the elements would help to centralize the flowerbed more (like the Arduino could technically just be a motor controller, or everything could be done by the Raspberry Pi with some extra tools to incorporate analog reading to the Raspberry Pi.

For physical elements, a lot of these can be improved by replacing the parts for better ones (as again, mentioned in the user manual/what we learned section below). The lights around the dial are using part of a test circuit from the previous group, and the connection for the e-break buttons can be improved as well. In case another group inherits the project, here are some pictures of measurements (there are some others in the flowerbed photos folder if you have access to that as well, but including some here for those who may only have access to the report)



These are measurements of the inside of the motor holder at different positions (with the breadboard in place to consider where to put the e-break buttons)

This is an extra light that can be soldered onto the dial (or elsewhere as needed) to make the connection better.

XI. Lessons Learned

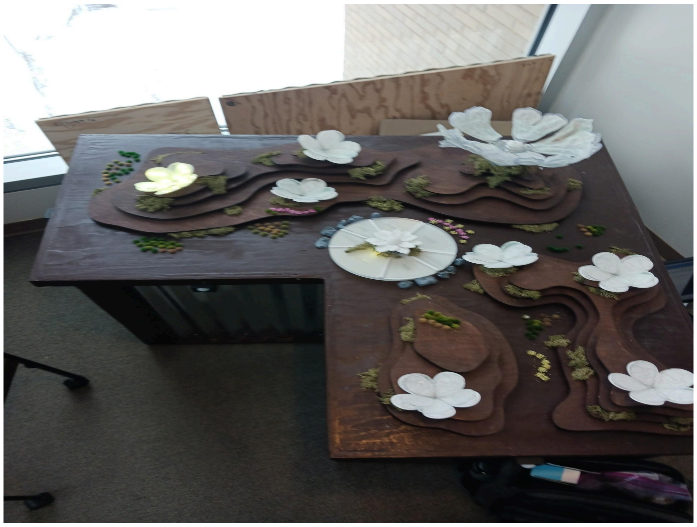
Although we believe we work well as a team, we still encounter some difficulties. Our biggest problem was learning how to pick up a project from another team. We ran into trouble learning where to start with how to fix the flowerbed from its initial given state. We learned to separate each component of the broken flowerbed and tackle each of them separately. Doing this prevented us from being overwhelmed with all of the problems and allowed us to make progress with fixing the flowerbed.

Another problem we ran into was communication. We had problems with finding the correct time to meet with our advisor, which prevented us from being able to get feedback during those weeks. We also had this issue with our client in the beginning of the project. We did not provide our client with frequent updates, which caused there to be a misunderstanding of the current state of the project at the time.

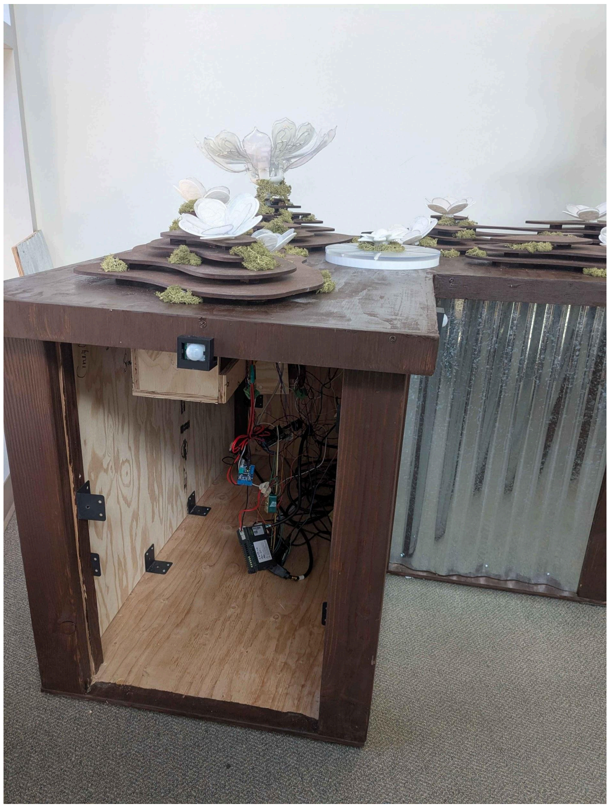
Below is a full user manual of sorts that details all of the different repairs we discovered throughout the process of repairing the flowerbed. Excluding replacement parts, you may need to have a screw driver (phillips and flat head) to do some repairs. <u>It is recommended that before making any changes, take pictures of the flowerbed so as to be able to recreate it in case something else breaks</u>:

- How to open the flowerbed

The top of the flowerbed can be lifted up and the walls on the sides slide up-after being unscrewed. We recommend shifting the lid a bit to get access to the two walls we removed, and try to pull them up.



This is a picture of the two walls on the side (the detached walls are the ones that normally belong on the left and right relative to this overhead angle)



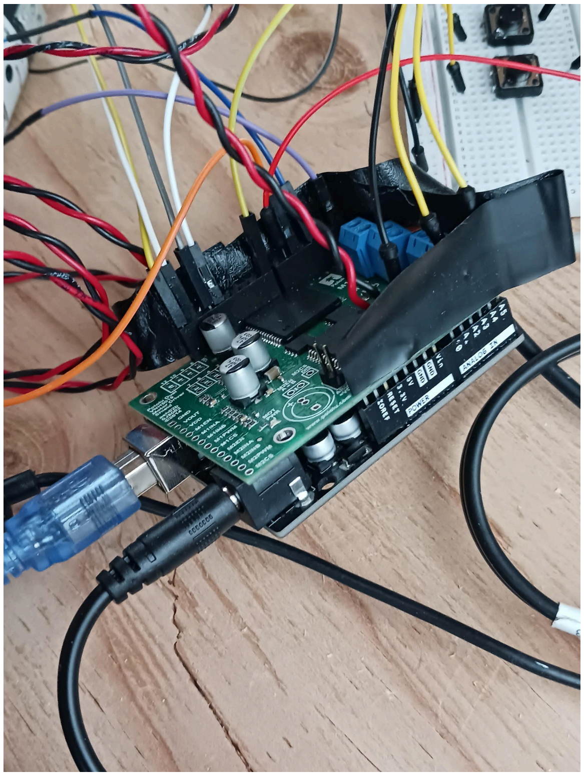
This is a picture of the smallest left wall, mentioned in the previous photo) removed, as of now the electronics are more positioned on the right side (again relative to the overhead view), so you also might have to shift the table a little to access that side.

- Are the lights not working

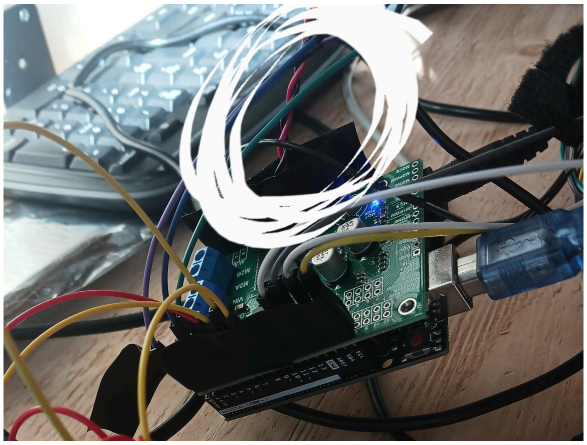
There are a couple of things to check if the lights aren't working and some of them are much harder to access, though depending on specifically how the lights aren't working, you can tell which of these are the problem. If push comes to shove, you can try them all, but would definitely recommend trying to fix the easier to access ones first. To figure out the specific problem, you can also use the led debug button. It should be the center one on the debug board and pressing it iterates through all the lights. Another general note for any solution, the lights are in a circuit together that is daisy chained below the table, each requires four wires in total: a red for voltage, black for ground (both give the lights power), and blue and white for indicating each light's number in the chain as well as which color it should be, all four need to be in to make the lights work. The one exception is that the lights around the dial are powered by the Arduino, while the lights for the flowers are powered by a separate cord that goes into the surge suppressor (note: by powered by we mean the lights will turn on due to those respective sources of power, the Arduino ultimately still controls which lights are what color, so it's possible that a light is getting power, but might not be told to turn on by the Arduino, though as you will see below, sometimes the lights have weird behaviour when they don't get enough power).

- None of the lights are turning on (flowers + dial)

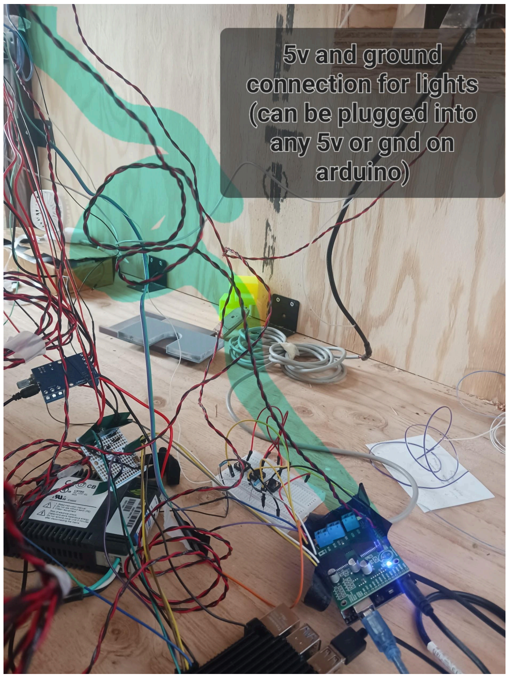
It's likely that the whole light system is not receiving power from the Arduino, the aforementioned red and black wires for the daisy chain start with the Arduino. Technically, you should be able to use any combo of a 5v and Ground source (even the ones on theRaspberry Pi) so if the connections are being used elsewhere, you can switch them. Though as of writing, this is what the connection looks like on the Arduino currently:



The two red and black wires on the Arduino, its a little hard to see with the tape on the way (if debugging feel free to take that off, its just kinda to keep the wires in place a bit, and kinda helps with the electrostatics a bit), but again these are connected to a 5v (for the red) and Ground (for the black).



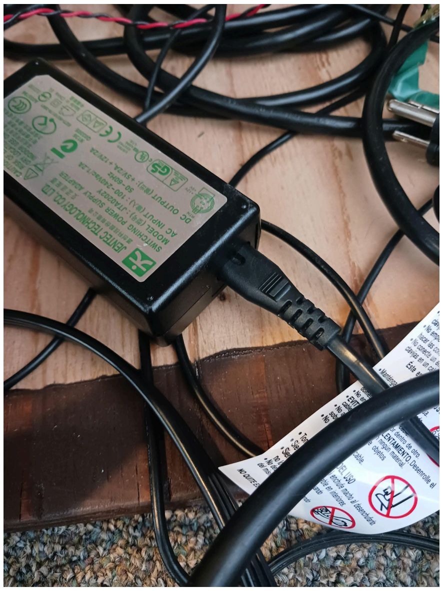
This is another angle of the two wires with them circled, from the other side in case the Arduino is rotated when you are investigating.


5v and ground connection for lights (can be plugged into any 5v or gnd on arduino)

This is the wire highlighted in reference to the whole inside of the table. If this wire falls out of the Arduino it can be a little hard to see as they will be likely dangling in the mass of wires (we have done some management, but its still a little hard, so apologies), but look out for two that are tied together. If you try this and it doesn't work, there is another section below for if none of the lights are turning on, although it likely indicates that there is a problem with the connections from the Arduino that are telling which lights to turn on, and not a power thing (keep reading the other solutions first as that one is the hardest).

- Flower lights are not turning on, but the lights around the dial are

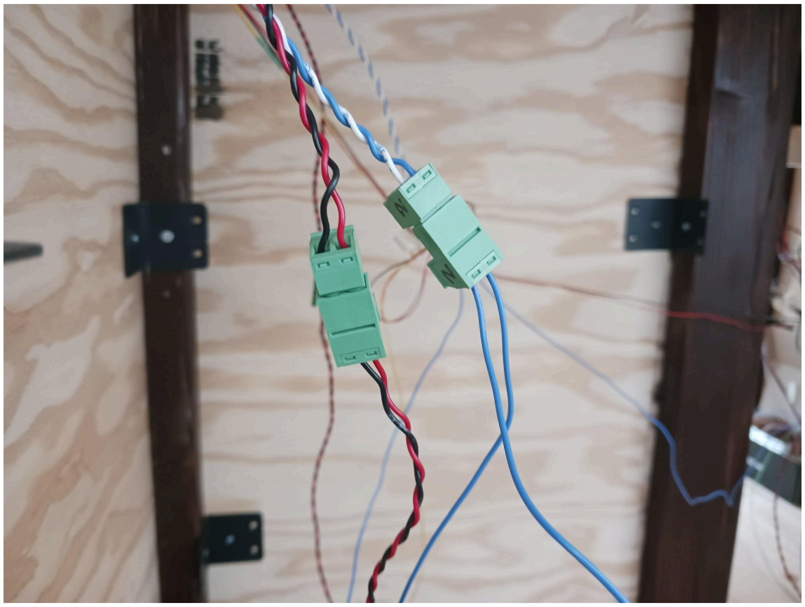
The lights for the flowers are powered by a cord that plugs into the surge suppressor, while the ones around the dial are powered by the Arduino. These cords are a bit messy, but labeled with tape (as of when writing this) on their ends. The main issue with this cord in particular is that this segment pictured below is a little sensitive and has a tendency to get detached, so aside from just checking that its plugged into the surge suppressor, I'd also check that this connection is made:



Like the other cords/wires, this can be a little messy, but I'd check for a lack of a connection on this box

- Only some of the flower lights are turning on

This likely means that the daisy chain broke (one of the red, black, blue, and or white wires that gets connected to the flower lights from below the table). Throughout the semester, none of these wires were broken, so it's more likely that one of these connections at the green bridges came apart. These can be kinda subtle, but again, are mapped to each of the flowers above, so if one of them is not turning on (and more specifically if the subsequent lights after this one are also not) check the daisy chain/bridges adjacent to this flower/the wires for issues.



By bridges I'm referring to the green connectors for the wires. These require a flathead screw to open up. Follow along the previous connection in the chain until you reach the first one that is not turning on, and see if there needs to be repairs.

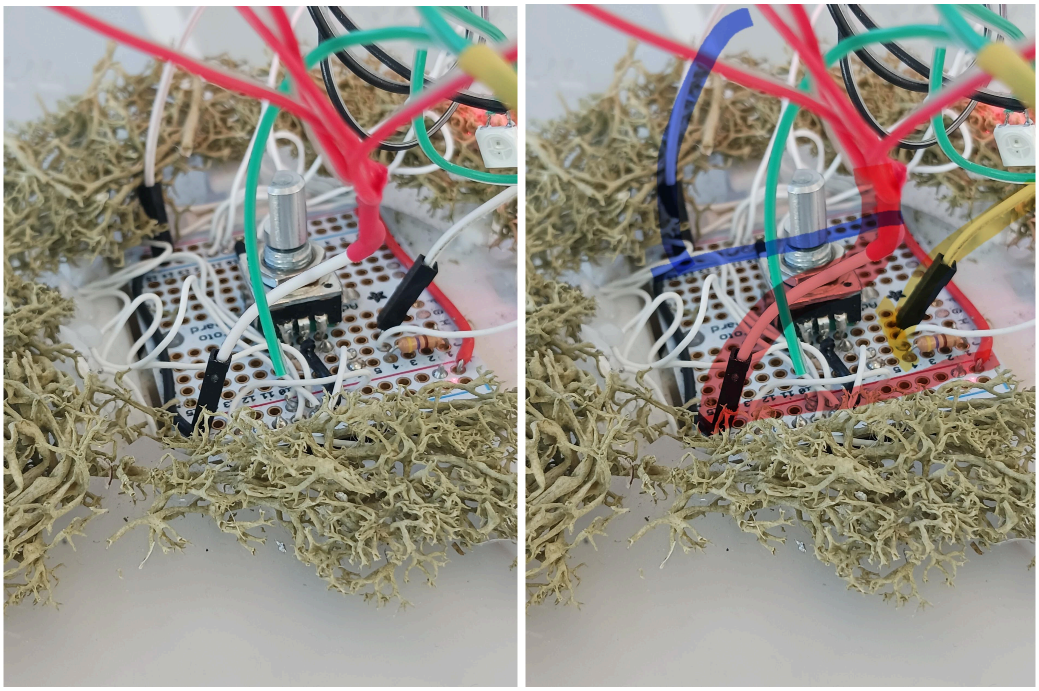
- None of the lights are still turning on OR the lights have weird patterns of which are on/what color they are

If you are more certain that the lights are receiving power, but not responding correctly, then the problem might be due to the Arduino not telling the correct lights to turn on. It is possible that this is a code thing, and if it is, back up the code that's currently on the Arduino, and try our code (specifically, press the debug button as that iterates through all the lights one at a time to let you investigate). If it does not appear to be a code problem then it might be a problem with the wires around the flower dial, as this is where the Arduino is first connecting to the daisy chain. To remove the flower dial, have one person hold the metal rotary switch (the circuit part that rotates) and another person gently lift the 3D printed flower up. There are four* main wires you will need to check,

- a red wire that connects to 5v (can be put into any of the holes aligned to the red column)
    - Try this one first if the lights are unchanging (this means if you are rotating the dial or have pressed the debug button and nothing is happening. You can also hook up a cord to the Arduino (ask Buddy/the labs downstairs for one) and put some prints in the Arduino code to make sure the lights should be changing)
- a black wire that connects to ground (can be put into any of the holes aligned to the blue column)
    - If the lights have weird behaviour (sometimes turning on, flickering, changing colors rapidly, its likely this one)
- a white wire that connects to the Arduino's pin that controls which light turns on (has to be in the row labeled 4 on the side that is farther from the motor flower, this should be the same row that one end of the resistor connects to)

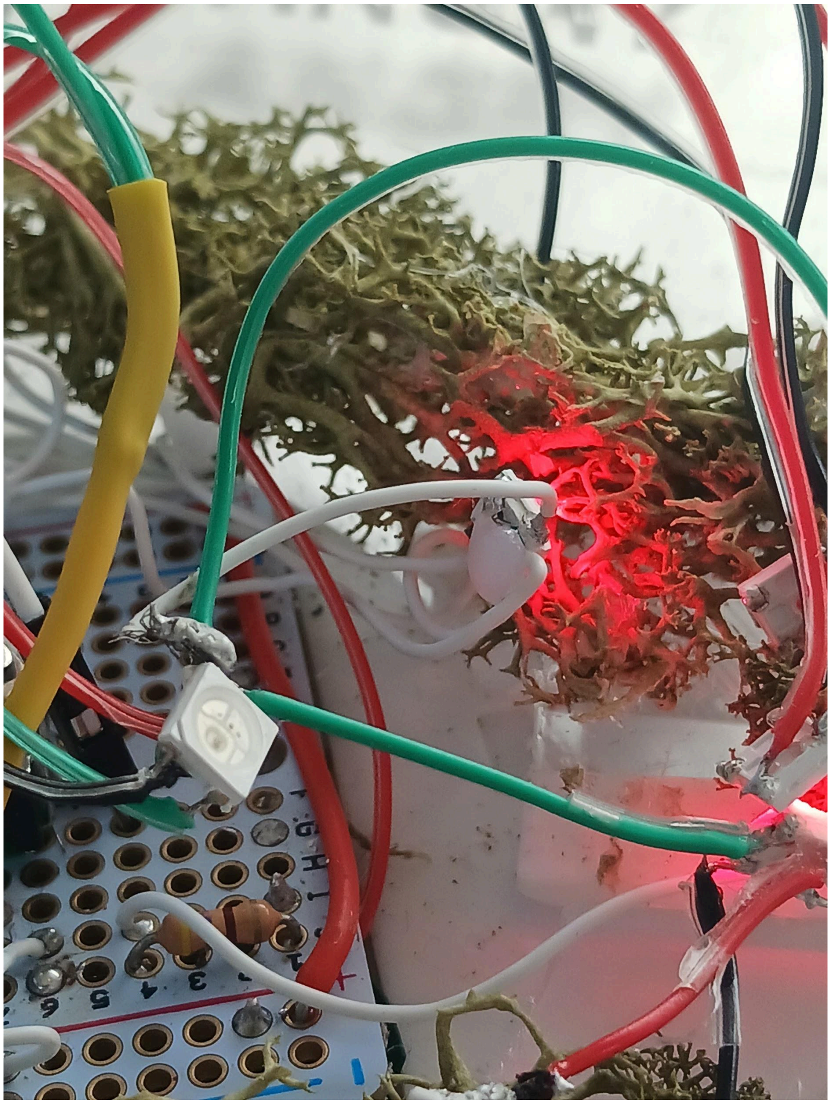
- (check for these wires last) and any of the wires in the chain that might have snapped, * there is one in particular that is likely the culprit, its located in the electrical tape that is wraps around two of the lights in the chain (and by two of the lights I mean there is a light that is wrapped up inside of this one since we didn't have a great way to resolder this back on, if you would like to my recommendation is to move the decorative moss out of the way and use a solder fan)

(these red and black ones power the lights, and continue it from below the table, even though the flower lights are powered from the cord and not the Arduino as mentioned above, all the lights are still connected with the daisy chain, so they need to communicate with each other, and if the ones in the start of the chain don't have power they can't do that)



This is a highlighted picture of the dial, with the red being the 5V, the blue being the Ground, and the yellow being the start of the chain (where the Arduino communicates with the LEDs). These are the first 3 of the wires to check mentioned above. This picture is a little outdated, these wires are all covered up by electrical tape, but still in these positions. The main thing to check is if these are touching the outside of the holes as this is not a traditional breadboard. Should you want to solder these in, I would be a little cautious and use the soldering iron and remove the moss from the sides. Also be aware of the fact that unsoldering is difficult especially with this being in the center.

If those 3 look good then it might be a connection between the lights themselves. Ideally there would be one light for each of the emotions on the dial, but due to a lack of resources, we were able to use a test circuit that the team before us had. What we ended up doing was attaching these two circuits together. Inside of the tape is another LED (connected in parallel to the one that controls the disgusted emotion, it also turns on when that one does). What you want to do (if you plan on reusing our circuit) is to connect the white wire that goes into the tape to the green one on the LED (this connects the two lights in the daisy chain). Doing this is very finicky (try to not use the tip of the white wire as we want as many of the strands to make contact with the metal on the green one), if you have enough time I'd consider soldering (again with all of the aforementioned warnings), or maybe even redoing the circuit to match the og diagram.

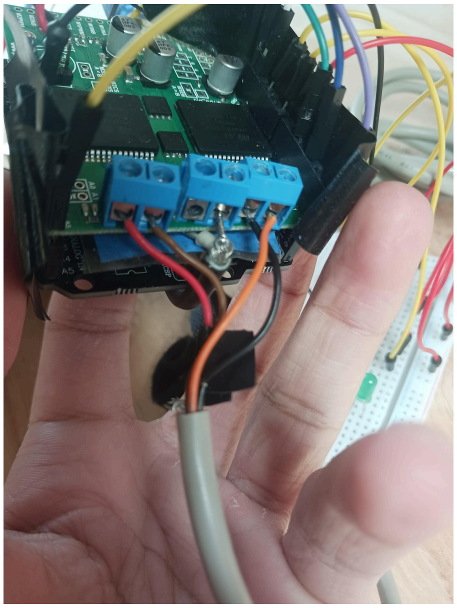


This is where the contact is (the sorta soldered part on the left), you can see the two lights in parallel are turned on right now.

*If you want a good look at the connection, and have the full drive that all of these are stored into, check out the LEDDaisyChainContact.mp4 video in the Flowerbed Photos folder.

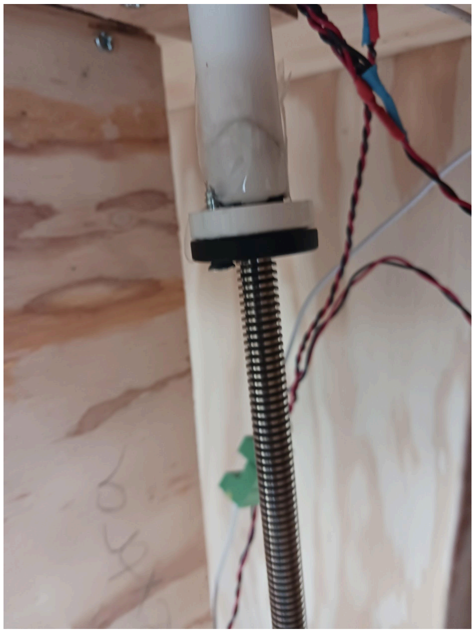
- Is the motor not working:
- The motor is not responding to anything (no noises or rotations or anything upon pressing buttons/code)

Check that the motor is plugged into the surge suppressor, and that the code is correct on the Arduino (details on what pins are used is included in that file). The other important area to check is the power on the Arduino's motor controller itself. This is located on top of the board and requires a flathead to access, I'd recommend taking a picture before unscrewing stuff like crazy, but here is our own picture if you want to reference it.



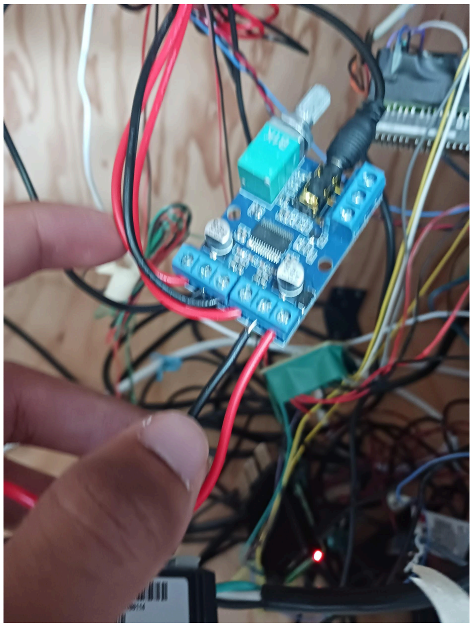
- The motor responds but is not great for lack of a better word

One thing we encountered with the motor is that while it might move, sometimes the flower doesn’t. Some of the physics for this were in the previous team’s document, but long story short, it's probably due to the motor itself being okay, but the flower being the part that’s disconnected. The best solution for this is hot glueing the pieces together as pictured below



- Is the audio not working:

Like some of the other problems, this likely just means something isn't plugged in/turned on. Make sure that the cable labeled "speaker" is plugged in. The other main thing to check is that the speakers are turned up, or rather the amplifier. Turn the dial on the end of it to make the volume greater (if you hear a click that means that it was turned off, try turning it the other way to make it go up).



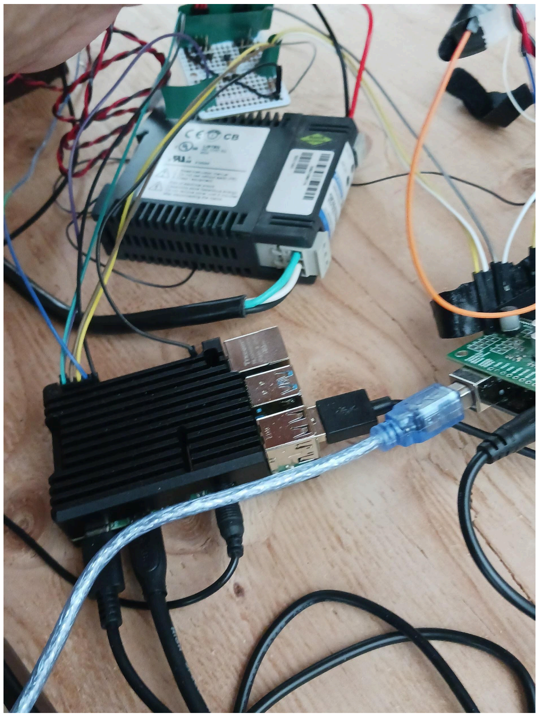
If both of those look good, the other area of concern would be if the Raspberry Pi is either not on, or more specifically is on and not communicating with the Arduino. The solutions to those problems are listed below in the website section (as the Raspberry Pi also communicates with the website).

- Is the site not updating

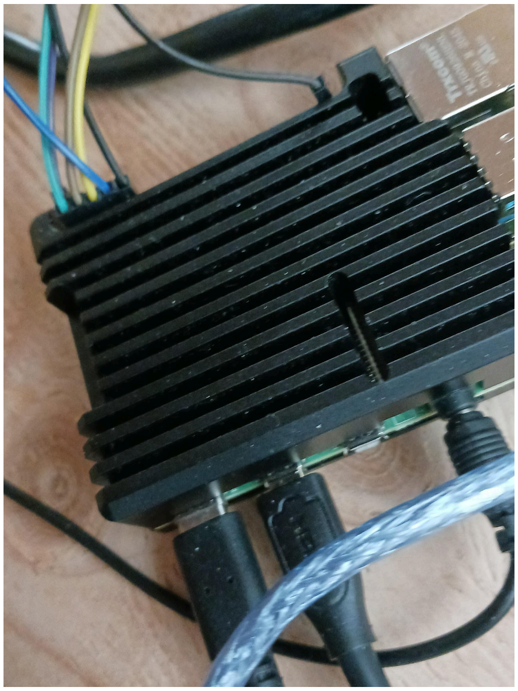
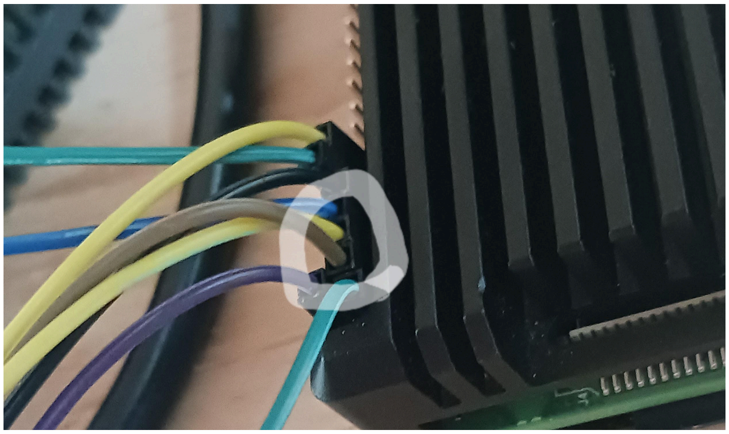
The Raspberry Pi has a light on the side of it, as long as it is plugged in, it should turn on by default when you initially plug it in and it receives power. However, if you would like to check if the code is working use the monitor. If you are not familiar with a Raspberry Pi, it's basically a desktop environment running a linux-like operating system called Raspberry Pi OS. If you plug in the hdmi cable to the monitor (also check that the monitor is on as it needs its own powersource) the pi should project its screen to it and you should see a desktop. Note that the pi we are using (Raspberry Pi 4) uses hdmi micro (not hdmi mini or regular hdmi), but the monitor uses regular hdmi (in case you need to get new ones). Just because the Raspberry Pi turns on, there is a chance that the program itself isn't running. You can open up a terminal by holding ctrl + alt + t. Here there are a couple things we want to check.

- <u>First, you can make sure the program is running on start up/not crashing</u> (if you are familiar with other classes it is the same deal, but incase you are not/need to know the file directories) * the biggest note for checking if the file is running on boot up is to navigate to the logging file located in "home/theflowerbed/Code/test" if the file is being updated on start-up, then the code is running. Details for navigating directories and running the code otherwise are below:
    - Try typing in "python 3 [directory_of_python_program]" and pressing enter, where the file is located in "home/theflowerbed/Code/real/mainV4.py". When you first open up the terminal, you are in the home directory, but the code references other files that are located in this directory, however to run the code on boot up, we use a file called a crontab, this runs code from the root directory, so to emulate running the code on boot up (while still being in the terminal to view any print statements) do the following upon first opening up the terminal (w/o the quotation marks):
        - "Cd /"

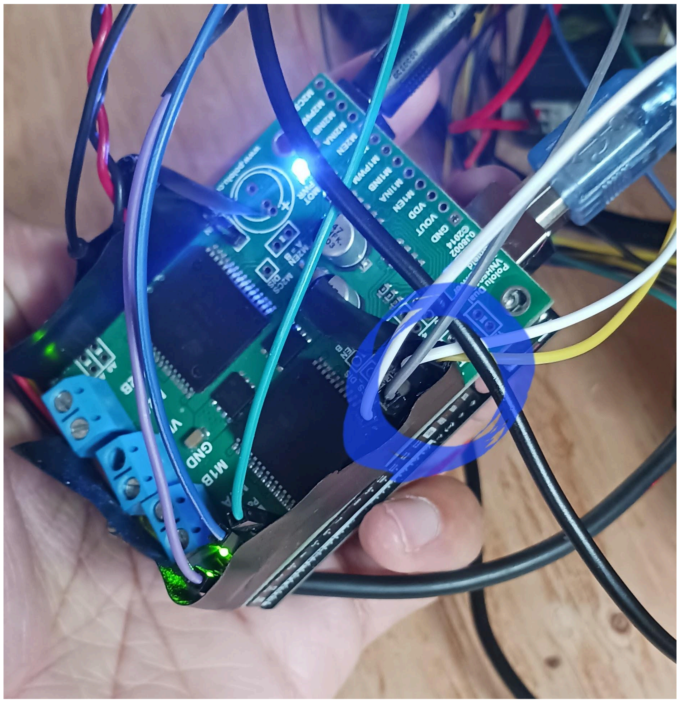
        - “Python3 home/theflowerbed/Code/real/mainV4.py”
    - If you would like to edit the crontab file itself type the following (doesn’t matter which directory you are in):
        - “Crontab -e”
            - At the bottom of this file (you can parse it by using up and down) is the code that runs the file on boot up. Ctrl + X lets you close out of the file, and with ctrl + Y lets you save it, Be careful about editing this as you can technically softlock the rpi (requiring a reflash of the OS), we didn’t run into that, but it is something you can technically do (if you are just running a python file are probably fine)
- <u>If the Raspberry Pi is on, but not communicating with the Arduino</u>
    - There are two wires to check on the Raspberry Pi and Arduino that are what connect them both (these are the I2C wires-all you need to know about these is that they let the Arduino and Raspberry Pi connect to each other and use the wire functions on both of the program’s respective code files). Try to match these pictures below (when we first received the broken flowerbed, these were swapped, so be careful of that)



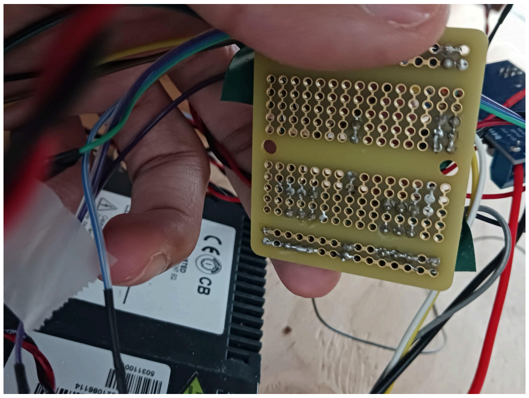
General gist is paying attention to the yellow and brown wires on the Raspberry Pi (left) and the white and yellow on the Arduino (right)

These are two closer pictures of the brown and yellow wires on the Raspberry Pi to pay attention to. Where the brown wire should be in GPIO 2 and the yellow in GPIO 3 (I'd look up a guide on all the button formats)



These are the two wires to pay attention on the Arduino. I realize that there are two white ones, the one relevant for the I2C connection is the left most one next to the yellow (these should be in the last two GPIO ports of the Arduino where the yellow is in port labeled "SCL" while the white is in the one labeled "SDA").



This last image shouldn't be super relevant, but this is the bottom of the board that connects the I2C from the Raspberry Pi and Arduino (again the biggest problem we encountered was making both of the wires be connected correctly on the Raspberry Pi and Arduino, but just in case you need a look at where the physical connection is made).

- <u>If the Raspberry Pi is communicating with the Arduino but not sending anything to the database</u>

You are going to need a bit more help for this one, especially if the Raspberry Pi is not connected to the internet. Let IT on campus know. As of writing, you need special permission to connect the Raspberry Pi to the wifi and this requires some codes that IT can get you. For what it is worth, in the Raspberry Pi login info doc, (in the resources folder) there is the password/key to "Mines Restricted" though if the problem is a lack of wifi connection, its likely due to that network being updated/no longer existing by the time your reading this. Otherwise if there are problems with access to the database itself, then check with the client as they should have ownership over this and can help with getting you in. There is an extra file or 2 in the home/theflowerbed/Code/real/ folder that has some stuff required for connecting to the database (this part should be okay, but just giving you a step in the right direction, or at least the one we used when hosting originally, pending on when you use this, this method might have been changed).

XII. Acknowledgments

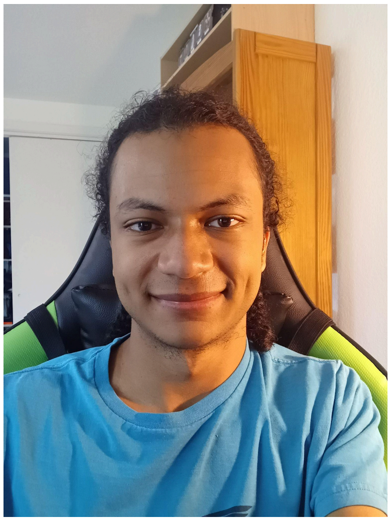
We would like to give special thanks to our client, Professor Iris Bahar, for entrusting and giving us an opportunity to work on this project. We would also like to extend our gratitude to our adviser Scott Jensen for his guidance and helping us learn better communication with our client.

We would like to thank the Mines IT department for their help with providing us the information we needed to connect our project to the university's wifi in a timely manner. We wish to acknowledge Arthur Lakes Library for providing the project with a keyboard and monitor and Professor Buddy Haun for providing tools whenever we did not have them on hand. We would also like to give thanks to the people who provided us with feedback on their experience with the website and the changes they would like to see.

Lastly we would like to thank our friends and family for being there with their support when we needed it throughout this entire project.

XIII. Team Profile

I'm Alex Morch, a senior in Computer Science, with a focus in Robotics and Intelligent Systems. I'm from San Diego, and have extensive experience in robotics, working on First Robotics teams for many years. I also have done plenty of coding projects, many of which are fun personal projects to explore an idea. I'm excited to work on the Emotional Flowerbed as it looks very creative and interactive.



HI, I'm Daniel Kenehan. I'm a senior majoring in Computer Science with a focus on computer engineering. I'm from Denver andI have some experience with hardware and software with small projects like tabletop arcade cabinets, computer repairs, and games. I'm interested in the project because it feels like we are working on an interactive museum exhibit, and making art.

I'm Angelina Mauleon from Denver and I am a senior in Computer Science with a track in Computer Engineering. I have experience in multiple coding projects ranging from using data from Excel sheets to automating machine testing. I am interested in the Emotional Flowerbed because I thought it was a beautiful interactive art piece that I wanted to help improve on. It's not too often that we get this opportunity of creativity in a computer science project.

Hello, I am Katie Lam, and I am a senior in Computer Science with a focus in Data Science. My hometown is Aurora, CO. I have some experience working with hardware and software from my Robotic team in high school. I chose to work on this project because I loved the interactive nature of the flowers moving and music being played to display emotions and how it brings more humanity to machines. My hobbies include spending time with my family and dogs and I enjoy reading in my free time.

References

Emotional Flower Project Final Report - Team 101-F24 RoboRenaissance

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations and acronyms

Term	Definition
LED	*a light-emitting diode*
URL	*Uniform Resource Locator, is the unique address used to specify the location of a resource (such as a webpage, image, or file) on the internet.*