Garden Pi – An automated home gardening solution

CSCI 43300

Austin Sands

[sandsa@iu.edu](mailto:sandsa@iu.edu)

ID# 2000139195

04/30/2023

Gardening at home can be a fun and rewarding hobby. However, some plants require constant care, and some schedules don’t lend themselves to the ability to give as much care to the plants as they may need. With the skills and knowledge gained during this course, I believe there is a way to use technology to help monitor and care for the plants without sacrificing much of your personal time. Utilizing a Raspberry Pi 3 and a few sensors, this project demonstrates how even a simple device that can handle some of the mundane gardening tasks such as monitoring light levels, checking if a plant needs water, and giving it water when it is required can help to keep healthy plants without much effort.

**Setup**

For the physical setup of the project, I am using the Raspberry Pi 3 Model B (referred to as Pi from now on). This project also utilizes a DS18B20 waterproof temperature sensor, a 5MP OV5647 camera model B0033, an ADS7830 8-Bit ADC, an HW-080 Resistive Moisture Sensor, a triple relay HAT module, a 12v water pump, a 12v solenoid valve, and lastly a 12v external power adapter. Aside from these electronic components, I have a jug to hold a large amount of water for the pump to pull from, a large tray to hold my plant and any overflow that may occur, and it all has been set in a place near a large window to get as much light during the day as possible.

The camera is simply plugged into the camera port on the Pi board with the included ribbon cable and requires no extra setup.

The relay hat plugs right onto the Pi header and all the GPIO pins pass through to the new headers on the relay HAT. Each of the three relays is powered by one of the GPIO pins by default, 26, 20, and 21 (BCM numbering used throughout).

The temperature sensor VCC is connected to 3v from the Pi, GND pin is connected to Pi ground, and DAT pin is connected to GPIO pin 4.

While the moisture sensor can be connected directly to power, constant power through the device is known to cause accelerated corrosion to the metal plating. Because of this, I decided to wire the sensor through one of the three relays to send power to the sensor only when desired. Therefore, 3v from the Pi is going to the common terminal of one of the relays (referred to as Moisture Relay). The Moisture Relay normally closed terminal is then connected to the VCC pin on the moisture sensor and the GND pin on the sensor is connected to Pi ground. Lastly, the DAT pin is connected to Pi GPIO pin 5.

The ADC module is an I2C chip and therefore must be connected to the proper pins on the Pi and I2C must be enabled in the Pi’s settings. The VCC pin of the ADC is connected to 3v from the Pi. The SDA and SLC pins are connected to the proper pins on the Pi, 3 and 5 respectively. GND pin is connected to Pi ground. Lastly, 3v power is supplied from the Pi and connected to channel 1 pin on the ADC with a 10k ohm resistor between the connection. Then a wire is connected from channel 1 to one end of a photoresistor and the other end of the photoresistor is connected to Pi ground.

Though the water pump used is rated for 12v power, it pulls and dispenses water much too quickly for the intended purpose, and so I found it was better to run at 5v, which the Pi is conveniently able to supply. So, the common terminal on the Pump Relay is supplied 5v from the Pi. The normally closed terminal is then connected to the positive lead of the pump and the negative pump lead is connected directly to a Pi ground pin separate from any previously used. This supplied the pump with enough power to turn on and pump water but at a much slower rate.

Lastly, a 12v wall plug is used with the positive lead connected to the common terminal of the Solenoid Relay. The normally closed terminal of this relay is attached to the positive terminal of the 12v solenoid valve, and the negative solenoid lead is connected directly to the 12v supply ground lead. This circuit is not directly connected to the Pi as it is not capable of handling the high voltage.

An image, diagram, and schematic of this setup is available in the appendix of this report.

**Software and Libraries**

To implement this project, I effectively turned my Pi into a LAMP server. Therefore, this project utilizes Python as my primary script coding language. I installed Apache2 so that I can host a webserver on the Pi to display and access the devices data from anywhere over the internet. MySQL is used (MariaDB specifically) to store the data gathered from the device in a database. Lastly, PHP was installed to access the database and display it on the Apache hosted website.

Along with Python the libraries that I used include: Gpiozero for I/O with GPIO pins, mysql.connector for writing information to the MariaDB database, Picamera2 (included with Bullseye OS for Pi) for the operation of camera, and w1thermsensor (also included with OS for Pi) for the operation of my temperature sensor.

**Operation**

First, an Apache webserver was created to host the website for the data to be displayed on.

Second, I created a database in MariaDB to store the data collected by this device. The database consists of four tables called temp\_data, light\_data, moisture\_data, and watering\_log. The first three tables all have the same format; The primary key is an auto-incrementing number starting at 1, then there is a datetime column, a column to store the actual data reading, and lastly a column called “manual\_read” which indicates whether this reading was from the automated script call or if a user requested this reading over the website. The data field for temp data displays temperatures in Fahrenheit, the light data is represented by a number from 0 to 255 where the lower the number the brighter the light is, and moisture data is represented as a text string of Wet or Dry. The watering\_log table has the same auto-incrementing primary key, a datetime, and a field called “manual\_call.”

Every hour on the hour, Chrontab runs the script I wrote for this device. First the script establishes a connection with the database created for this project. Once connected, the script will take a picture using the attached camera unless the time is between 8pm and 7am (too dark to notice anything in the pictures). This picture is saved using the datetime that the script was run. A copy of this new picture is made with the title “recent.jpg” and both are saved in the “/var/www/” directory for this Apache webserver. This copied image is effectively overwritten every time the script is run so that “recent.jpg” always is the most recent picture taken. Next, the Moisture Relay is powered on so that power is now being sent to the moisture sensor. Then a temperature reading is taken and saved to a variable. Likewise, both light levels and moisture levels are also taken and saved to variables. After these readings are taken, Moisture relay is then powered off. During that time, if the moisture reading is determined to be dry, then the script will power on the Solenoid and Pump relays to give the plant water for 5 seconds. Lastly, the script writes all this data to the appropriate tables in the database and commits the entries. The only thing that the user needs to do in this process is check the water container every so often to ensure there is always a supply of water for the pump.

Aside from the automatic operation, users can access the Apache webserver to see the data gathered by the device. The landing page of the site displays the most recent image as well as the most recent reading for each respective category and the last time that the plant was watered. Under each of the data displays, there is also a button that when pressed will run a separate script for each button that either gets a current reading for that datapoint or waters the plant. Atop the page is a navigation bar with links to pages focusing on each specific data point. Clicking on these links will send you to a page that, in the case of the gallery displays ten images at a time in a paged format, or in the case of temp/light/moisture data, shows a table of 25 entries per page sorted by date of entry.

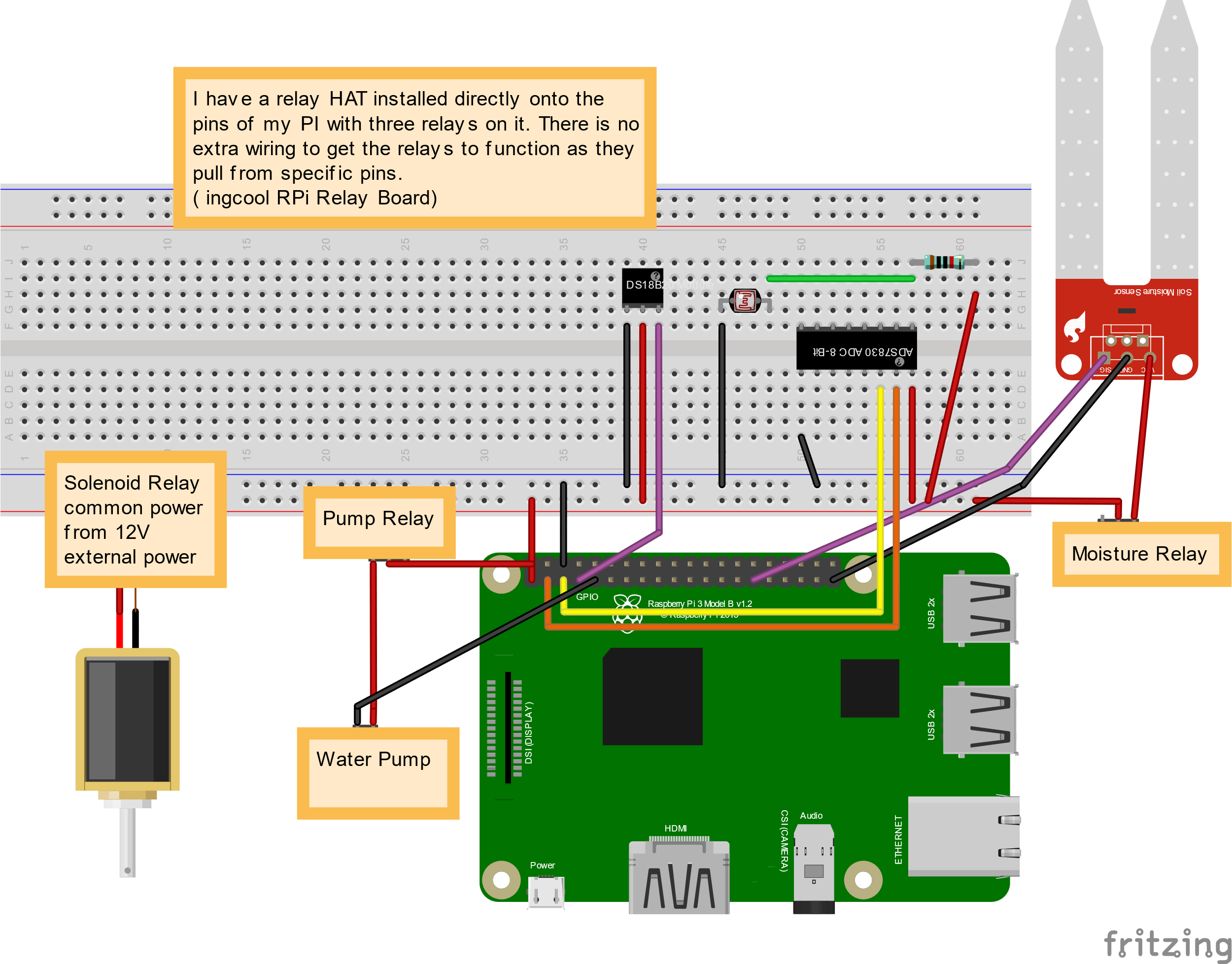
There are images included in the appendix of some of the website pages.

**Difficulties and Obstacles**

The only real difficulties that I encountered during the implementation of this project were hardware issues. The initial pumps that I wanted to use only worked for about a day of being submersed and then would fail, so I needed to get a new one. Other than that minor issue it just came down to learning how to integrate all these different platforms into one cohesive project and it just took a little bit of time.

**Appendix**

****Project setup

Diagram of Setup

SchematicDiagram, schematic

Description automatically generated

Wireshark

A picture containing text

Description automatically generated

Wireshark file included with project submission, filter with HTTP tag to see site communication