LIRMS Manual

*Udaya Kothapalli (*udaybhaskar.k@tamu.edu*)*

lirms.tamu.edu
September 1, 2017

Contents

1 INSTALLATION WORK FLOW

The following are the steps to build and test LIRMS:

1.1 Field Side Hardware Installation

1. Purchase all the required Hardware Components. **(refer: 3.1 and 4.1)**
2. The controller unit and the GSM modem have to be updated with the latest firmware to have a smooth functioning of the system. Please follow and complete Hardware Setup section. **(refer: 3.2)**
3. After the hardware setup, follow the Hardware Connections section and perform all connections. **(refer: 4.2)**
4. By now most of the hardware setup is completed, for updating the LIRMS software on the controller, please follow and complete the Software Setup section. **(refer: 3.4)**
5. Boot up the GSM modem by following the below steps:

   Insert the SIM card to module and lock it.

   Connect the adapter to module and turn it ON!

   Now wait for some time (say 1 minute) and see the blinking rate of "status LED" (GSM module will take some time to establish connection with mobile network)

   Once the connection is established successfully, the status LED will blink continuously every 3 seconds
6. Thats all on the hardware side!

1.2 Server Side Installation

1. Purchase all the required Components to setup a server. **(refer: 5.1)**
2. Now install all the dependencies on the Intel NUC. **(refer: 5.2)**
3. Now deploy the server logic and user interface. **(refer: 5.3)**
4. Follow and complete the section on Downloading Weather Data. **(refer: 5.4.3)**
5. Once the above steps are completed, most of the server setup is completed and ready to control/program the field side controller. Please follow the subsection on Programming Irrigation Cycle. **(refer: 5.4.5)**

6. For visualizing the irrigation cycle, please follow subsection on Visualizing Irrigation Cycle. **(refer: 5.4.6)**
7. Thats all on the server side!

2 HARDWARE

2.1 Purchase List

Please find the Purchase list of all the hardware components in the excel sheet found at: https://www.dropbox.com/s/1hek61wcsi6ssfh/2nd%20Generation%20Purchase%20list%20w_o%20tablet.xlsx?dl=0

3 CONTROLLER

The following section explains about the setup and programming of the irrigation controller.

3.1 Hardware Components

For constructing an irrigation controller we will need the following components shown in the table ??.

1. Intel Galileo Gen-2: https://goo.gl/7Nt8aQ
2. 16 GB micro-SD card: https://goo.gl/K9Vrj4
3. 20mm Coin Cell: https://goo.gl/D34n5L
4. CR2032 Lithium Coin Cell Battery: https://goo.gl/GKkrya
5. GSM/3G shield for Arduino:
6. T-mobile SIM card: Unlocked, GSM, Quad-Band 850/ 900/ 1800/ 1900 MHz
7. Soil Moisture Sensor: https://goo.gl/XF2P3C
8. Micro-USB to USB connector
9. FTDI Serial TTL-232 USB Cable: https://goo.gl/5XG1Fe

3.2 Hardware Setup

The following section explains the initial setup of the hardware components.

3.2.1 Intel Galileo: Updating Firmware

While building a new controller, one should update the firmware on the Intel Galileo with the latest version to make sure serial and other functionality works properly.

Download the firmware from https://downloadcenter.intel.com/download/26417?v=t.

Please follow the following web tutorial to update the firmware on the Intel Galileo. https://downloadmirror.intel.com/26417/eng/Galileo_FW_tool-UserGuide.pdf.

3.2.2 Intel Galileo: Installing Operating System

Intel Galileo has an inbuilt Operating System right out of the box. However the problem with this is observed when an arduino

sketch/program is lost once the device loses power. To avoid this sketch persistence issue, we need to install an operating system using a micro-SD card.

Please use the following web tutorial to install Operating system in Intel Galileo. `https://software.intel.com/en-us/get-started-galileo-windows-step1`. Installing Operating System using an SD card is important because by doing so we can enable persistent arduino sketches.

3.2.3 GSM modem: Updating Firmware

Similar to the firmware update that was performed for Intel Galileo, a firmware update is needed for the GSM modem as well. Based on the SIMCOM chip installed on the modem, follow the corresponding tutorials.

1. SIMCOM SIM900A modem: `https://goo.gl/Db32Ay`
2. SIMCOM SIM5320A modem: `https://goo.gl/YiXBZT`

Anyways, the below describes the connections needed to update the firmware on the GSM modem.

1. *TXD* (SIM900) $\longrightarrow$ RXD (RS232)
2. *RXD* (SIM900) $\longrightarrow$ TXD (RS232)
3. *GND* (SIM900) $\longrightarrow$ GND (Supply or from Arduino)
4. *4.2V* (SIM900) $\longrightarrow$ 5V (Supply or from Arduino)
5. *GND* (SIM900) $\longrightarrow$ GND (Supply or from Arduino)

3.3 Hardware Connections

3.3.1 Connecting RTC

Intel Galileo Gen-2 has an inbuilt RTC to keep track of time. This RTC works when the board is powered and the time resets to default value once the power supply is stopped. To avoid this we use a 20mm Coin Cell to power the RTC continuously.

The figure 1 shows the power pins to connect the 20mm Coin Cell to power the RTC on the board.

The following shows the connections of the Coin Cell with the RTC power pins on the board.

1. + terminal on board $\longrightarrow$ *ON* terminal on the Coin Cell
2. - terminal on board $\longrightarrow$ *GND* terminal on the Coin Cell

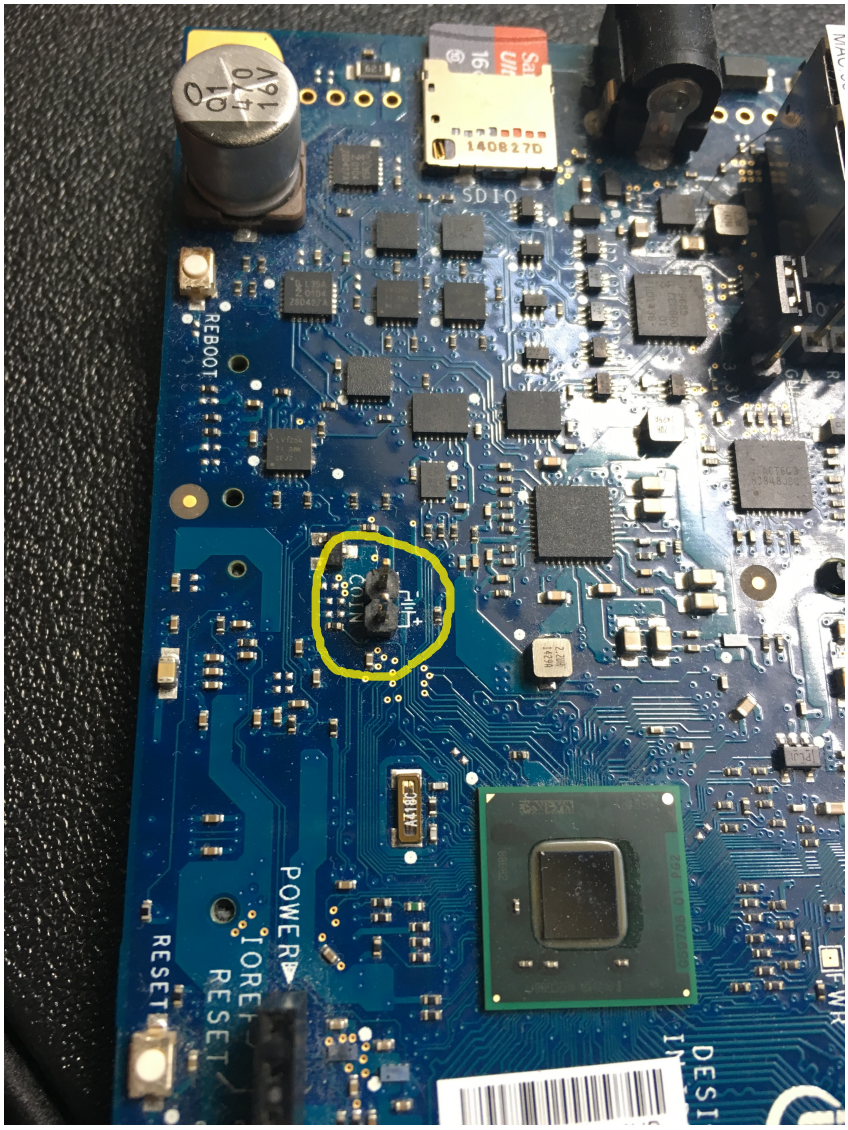


Figure 1: RTC power pins

3.3.2 Connecting GSM modem

GSM modem is used for enabling internet connection to the controller. A T-mobile SIM card is used for providing the network. An unlocked T-mobile SIM is used. First insert the SIM card into the SIM card holder of the modem as shown in the figure 2. Connect the antenna provided to the modem.

Most GSM modems are compatible with Arduino type prototypes. Therefore, it is simple to align the PINs of the modem with that of the Intel Galileo to mount the modem on top of the Intel Galileo. After mounting the modem onto the Galileo should look something similar to as shown in the figure 3

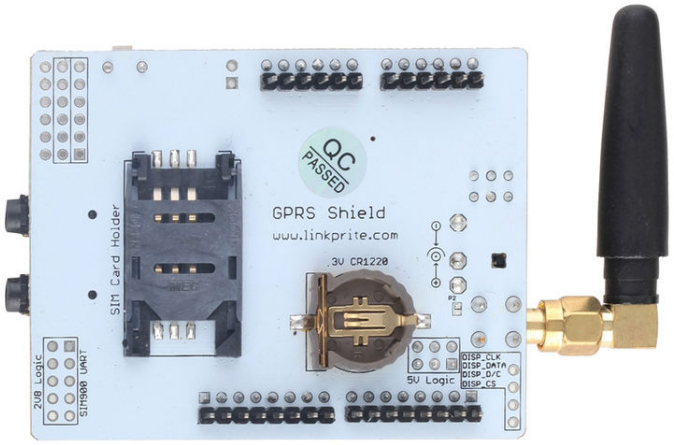


Figure 2: Back side of the GSM modem

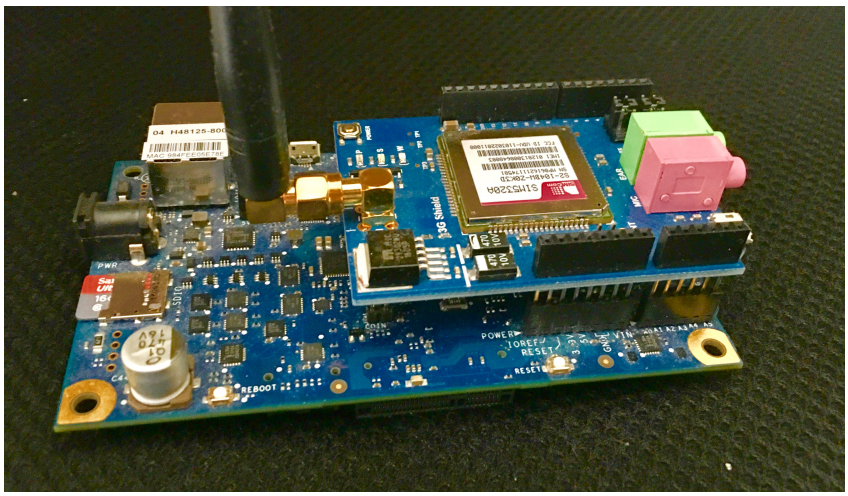


Figure 3: GSM modem mounted onto Intel Galileo

3.4 Software Setup

3.4.1 Installing Programming Environment/Compiler

1. Please follow the link to download and install the compiler.
   `https://www.dropbox.com/s/6ddktojzpxxv36m/IntelArduino.`

zip?dl=0

2. After download extract the compressed "IntelArduino.zip" file.

3. When you check the folder "IntelArduino". You should find the following files as shown in the figure 4.

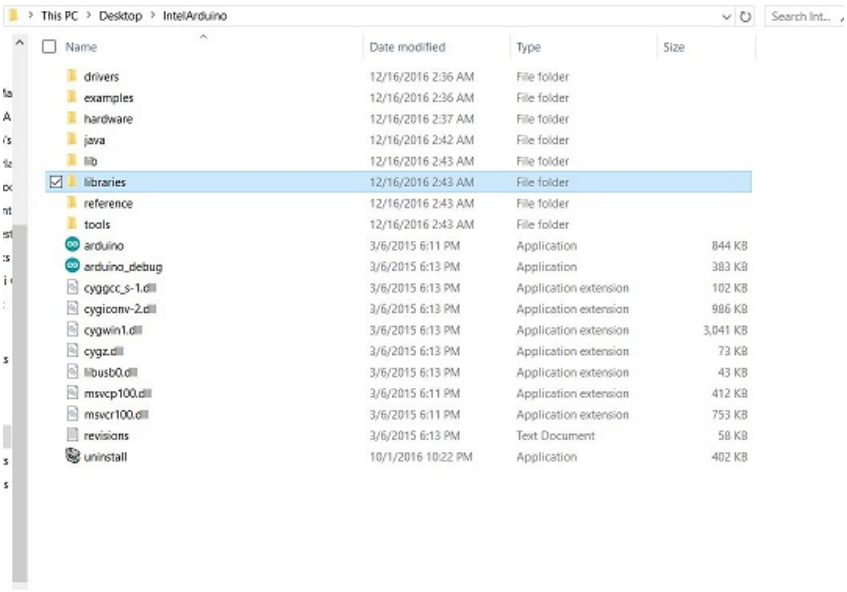


Figure 4: Arduino Folder

4. Now move the uncompressed "IntelArduino" folder to your Windows "C: " drive. **Double click on "arduino" application file shown in the figure 4. This will launch the Arduino IDE**.

5. If prompted for a firewall access by the windows security, Select allow access (Don't worry It is secure!!). Check the figure 5 to know how the pop-up looks like.

3.4.2 Downloading LIRMS controller software

1. Please download and extract the Dropbox file available at this link.

2. The uncompressed folder "z_main" should have five files.

3.4.3 Setup to program Arduino

Hardware Connections

1. After the first two steps you will have the compiler(IDE) and the LIRMS software ready for installation. This last step is to program the controller board.

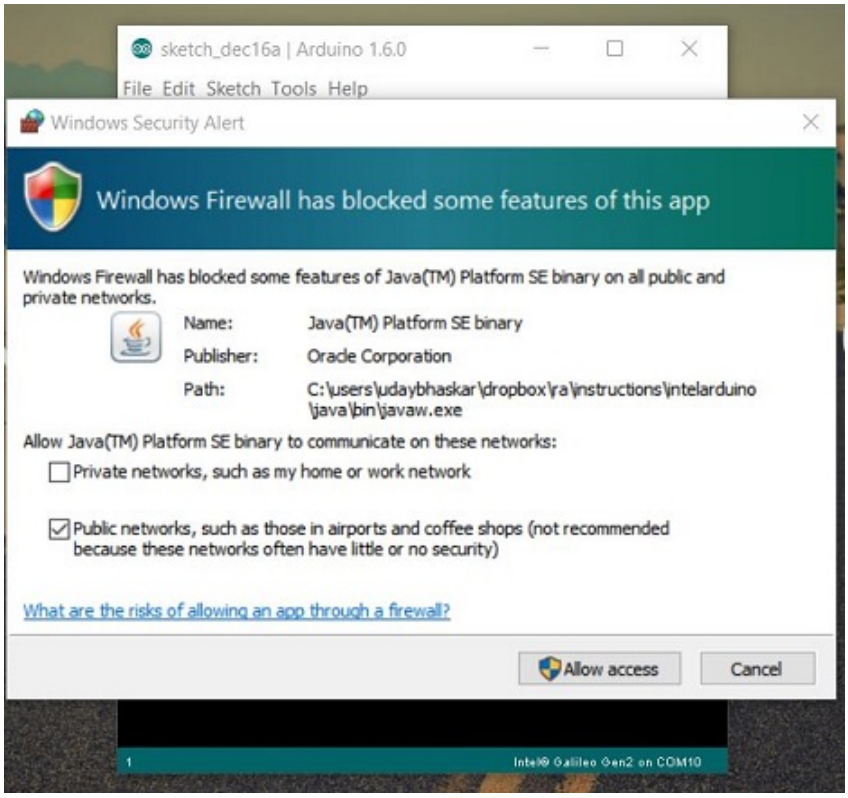


Figure 5: Firewall permission popup

2. Plug in the Power supply to the controller power socket as shown in the figure 6.
3. After powering up the controller, you should see at first three "GREEN" LEDs light up. As shown in the figure 7.
4. After few seconds, you should see a fourth "GREEN" LEDs light up. As shown in the figure 8.
5. Now at this point you can go ahead and connect a USB cable to the USB port on the controller board as shown in the figure 9. (You need a micro USB cable - most android phones have this cable).
6. Now you can connect the other end of the USB cable to your computer's USB port's as shown in the figure 10.

3.4.4 Updating/Programming Controller

1. Launch the Arduino IDE and open the "z_main" arduino file. It should look like shown in figure 11.
2. Now we have to select our controller board from the "Tools" menu as shown in the figure 12.

Figure 6: Power connection

3. Now we have to select the "Port" number to which our controller is hookedup to our computer. It would change from computer to computer . As shown in the figure 13, it was "COM5" for me. You might have a different "COM" port. Select your port.

4. And this would be the last settings that you will configure. This is the programmer selection. Follow as shown in the figure 14 and select "ArduinoISP".

5. After all the configurating steps, now click on the "Tick mark" button below the main menu. That will start the compilation process. Wait till you see a "Done Compiling" message on the bottom of your IDE as shown in the figure 15.

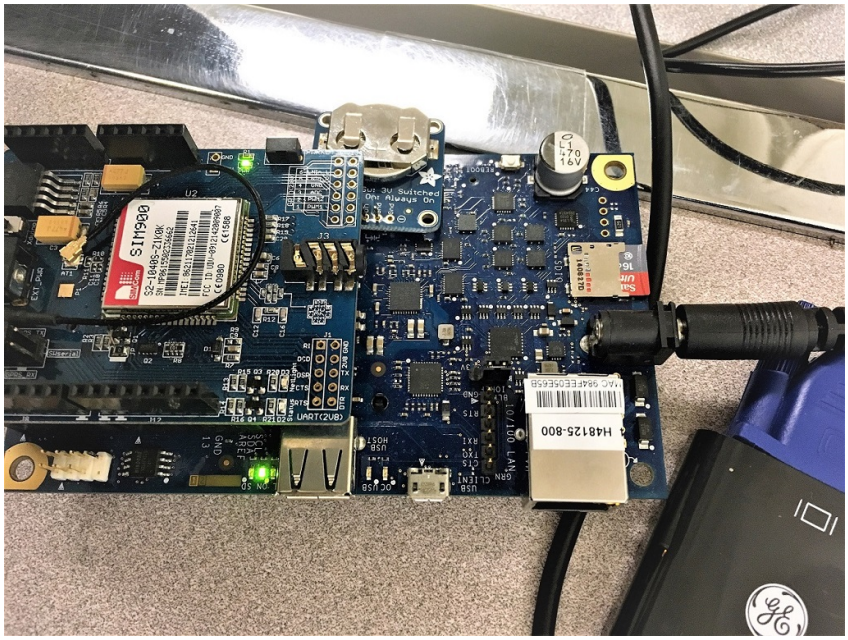


Figure 7: Powerup initial

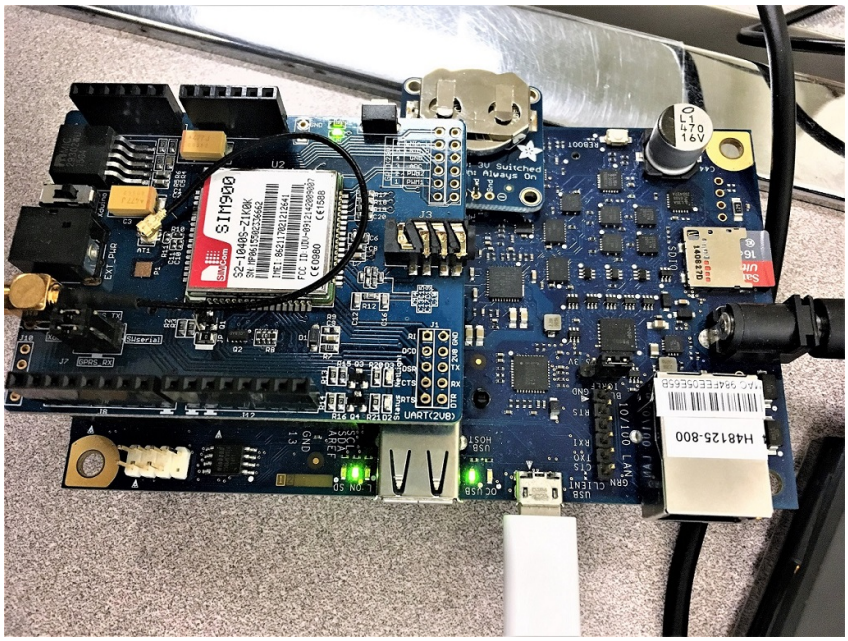


Figure 8: Powering up: Complete

6. Once the compilation is done, uploading the program to the controller is left. To upload the program, click on the "Right Arrow" button next to the compilation button. Once the button is pressed wait till you see a "Done Uploading." message as shown in the figure 16.

Figure 9: USB connections to Controller

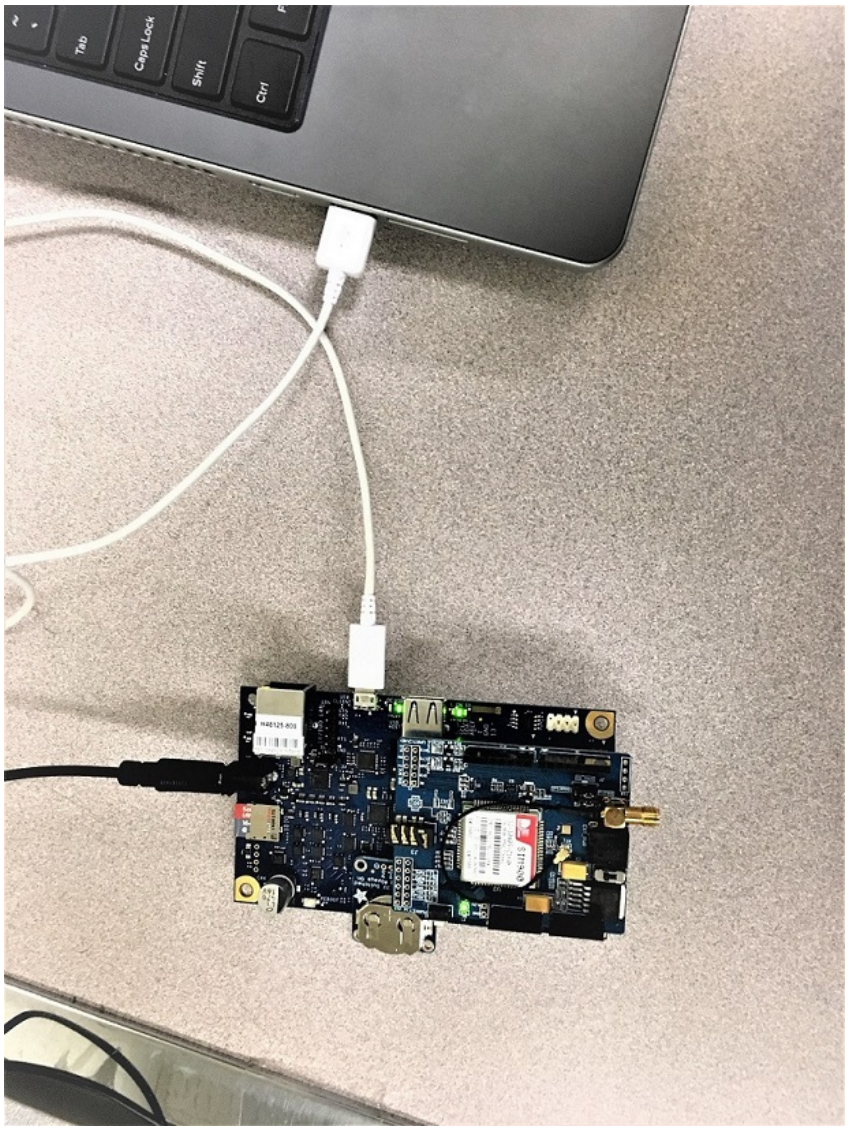


Figure 10: USB connections of Controller to Laptop

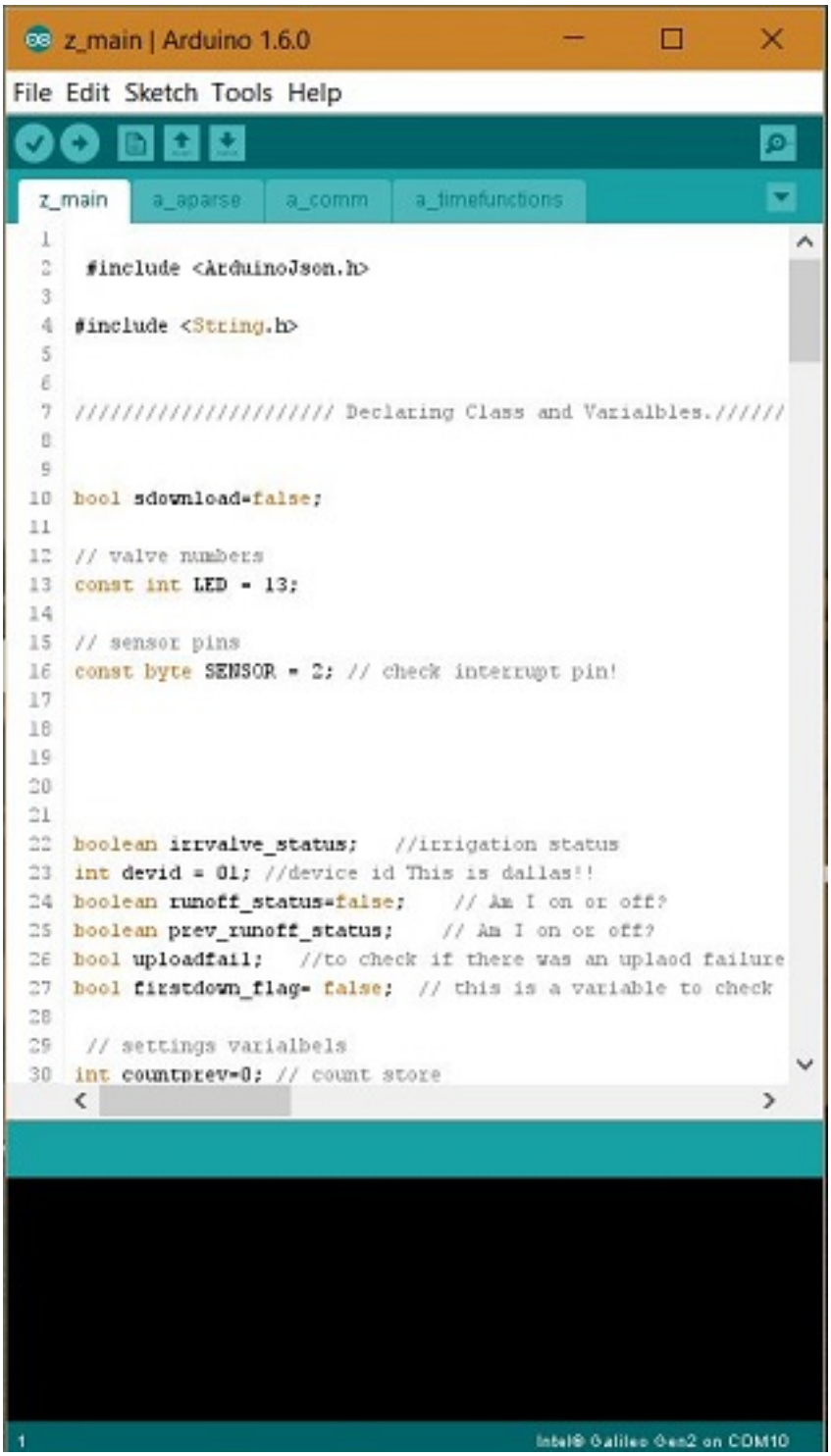


Figure 11: IDE start screen

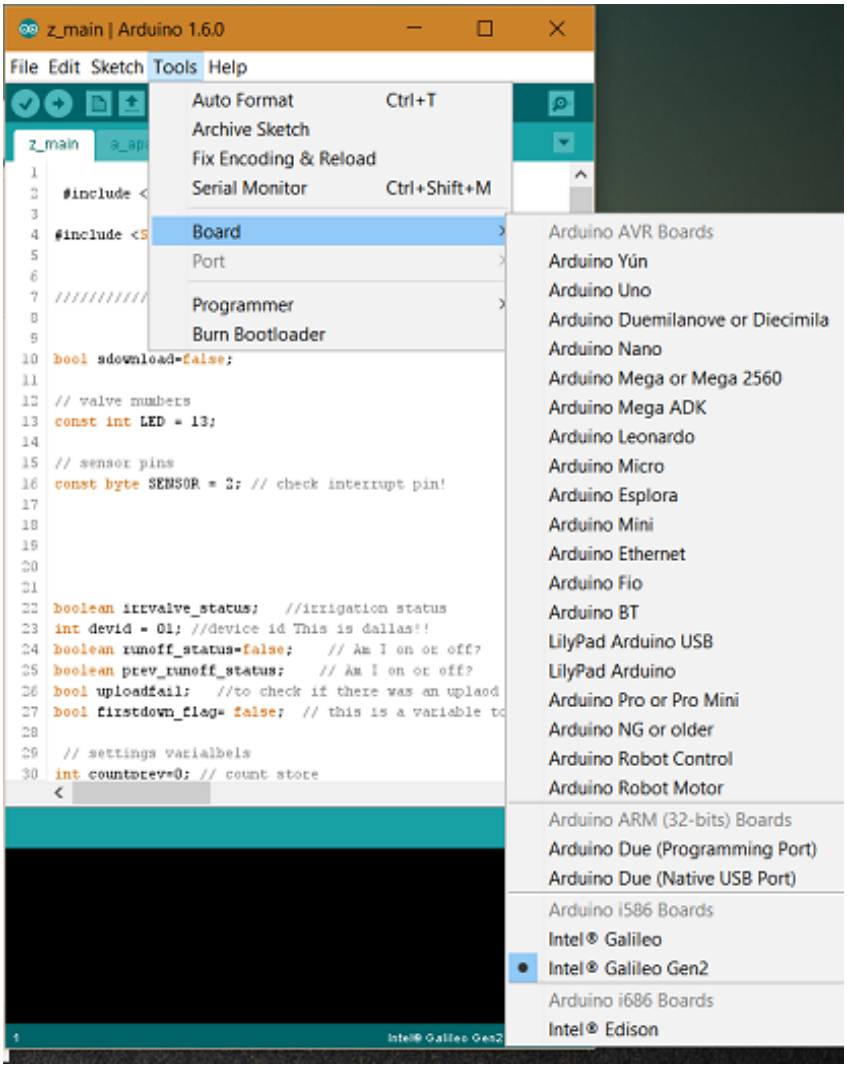


Figure 12: Board Selection

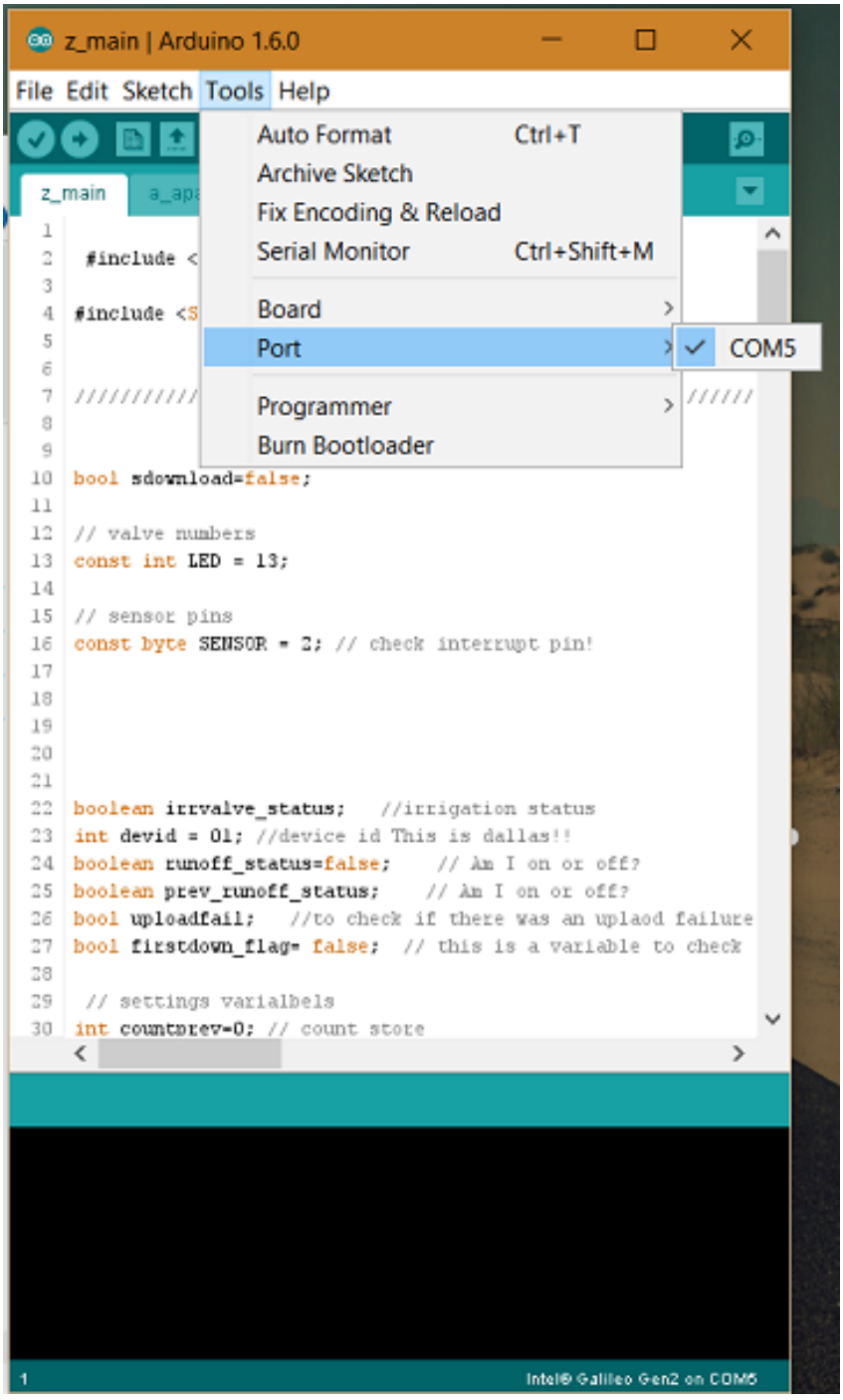


Figure 13: Port Selection

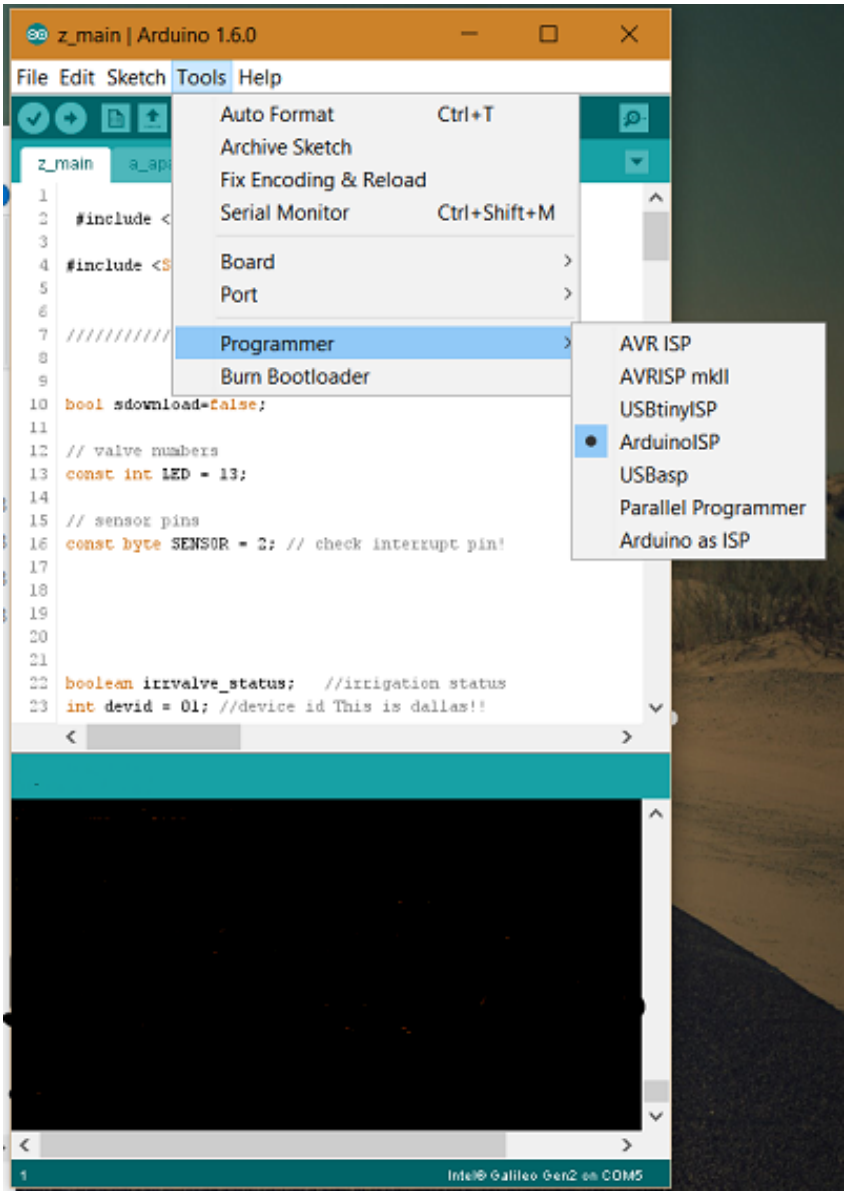


Figure 14: Programmer Selection

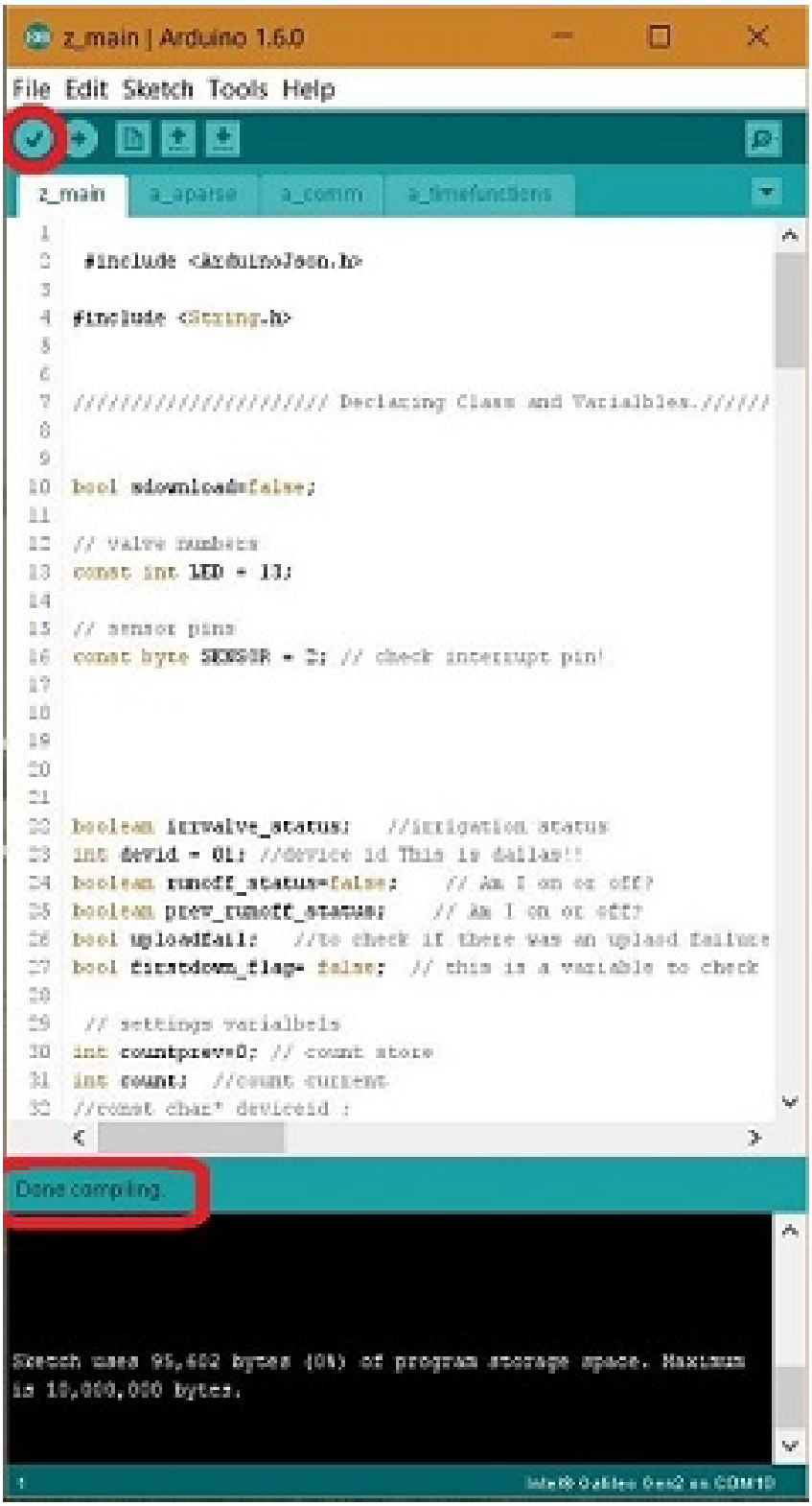


Figure 15: Done Compiling

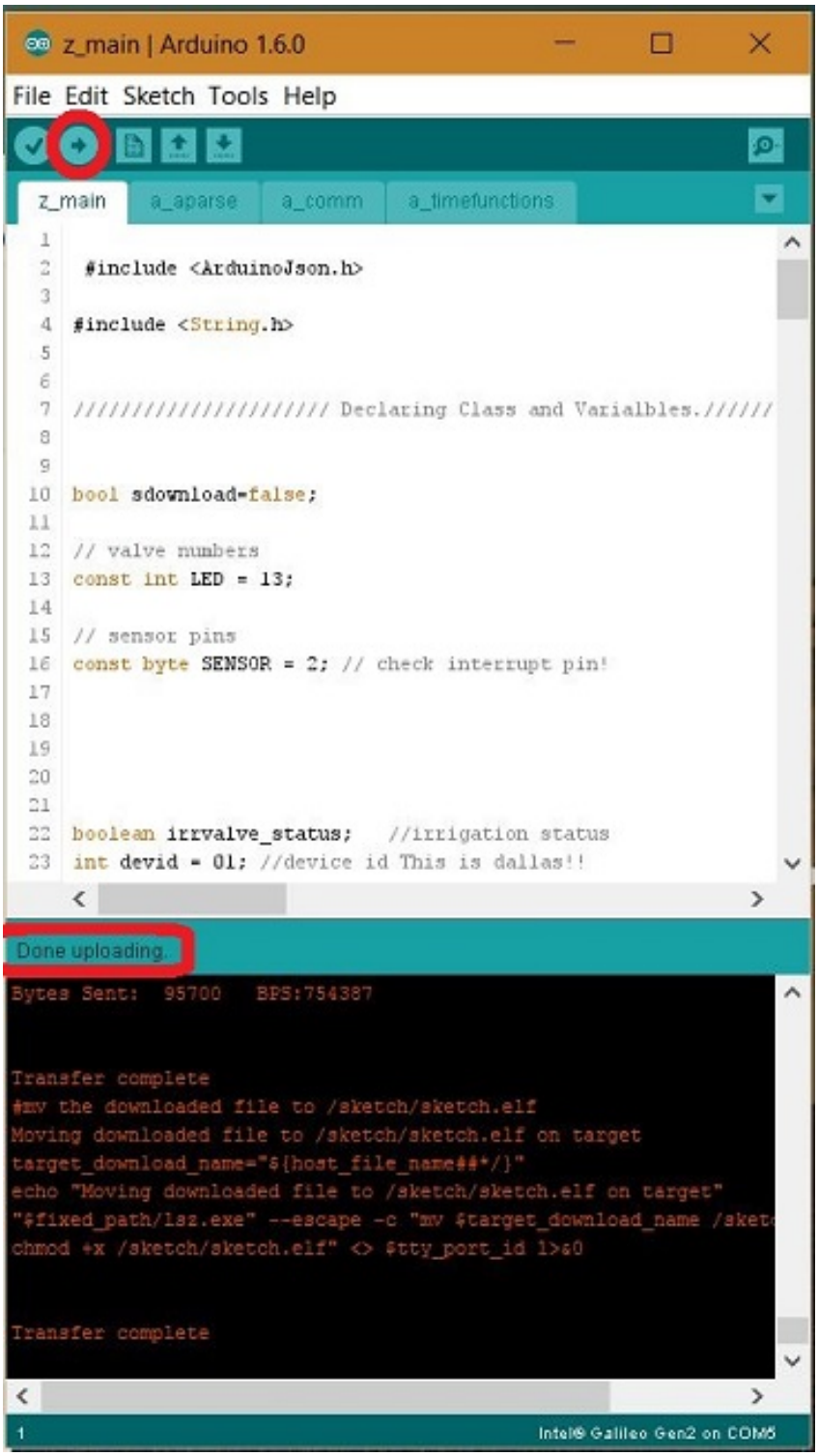


Figure 16: Done Uploading

4 LIRMS

4.1 Hardware Components

The following are the components needed for building a working LIRMS irrigation system.

1. Relay Module: `https://goo.gl/oirgHd`
2. 24V AC adapter: `https://goo.gl/wYu7ZT`
3. Irrigation wire

4.2 Hardware Connections

4.2.1 Connecting Relay Module

Each irrigation valve needs one relay to control the irrigation cycle. The figure 17 shows connections for one relay, this can be extended to multiple relays.

Figure 17: Relay module connections

In the implementation, both the *control* and *LIRMS* plot's valves are controller using the *Relay-6* and *Relay-4* respectively. The following describes the Pin connections.

1. **(LIRMS)** *Pin-5* on Controller $\longrightarrow$ *IN4* on Relay Module
2. **(Control)** *Pin-6* on Controller $\longrightarrow$ *IN6* on Relay Module
3. *GND* on Controller $\longrightarrow$ *GND* on Relay Module
4. *5V* on Controller $\longrightarrow$ *VCC* on Relay Module

4.2.2 Connecting Runoff Sensor

For connecting the Runoff Sensor to the Controller, two wires from the sensor are connected to *GND* and a *Digital Pin* on the controller. For the implementation, *PIN 2* is connected to the runoff sensor along with *GND* pin from the controller. Figure 19 shows the Pinout diagram for the controller.

The below describes the pin connections between controller and runoff sensor.

1. **(LIRMS)** *Pin-2* on Controller ⟶ *Red wire* on Relay Module
2. *GND* on Controller ⟶ *GND (Black wire)* on Relay Module

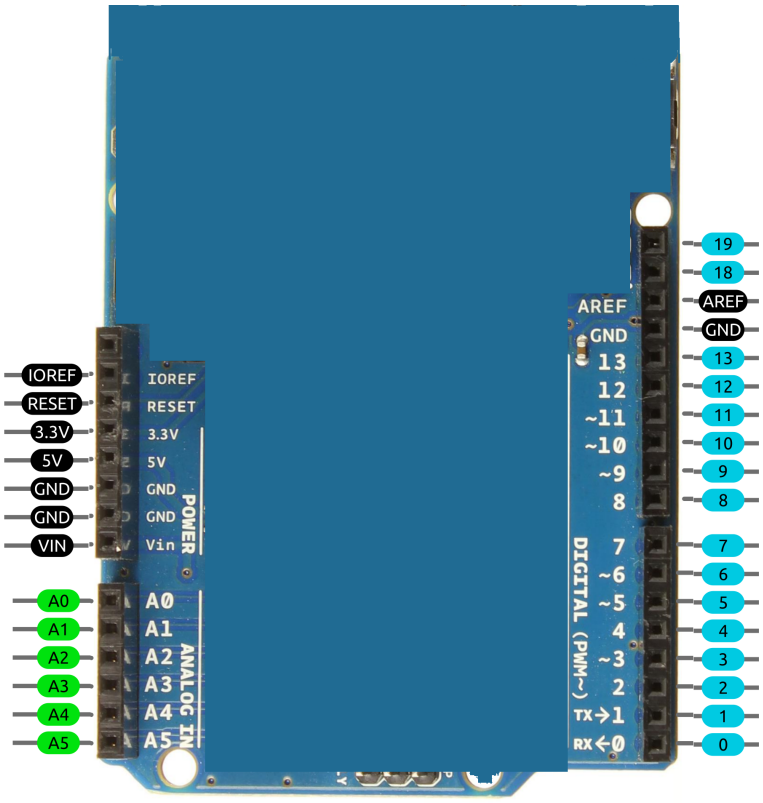


Figure 18: Arduino Pinout

4.2.3 Connecting to the Irrigation Valve

For connecting the Irrigation valves to the relay module follow the connections shown in the figure 19.

The following describes the connections:

1. First terminal of 24V AC $\longrightarrow$ First terminal of Irrigation valve
2. Second terminal of 24V AC $\longrightarrow$ *COM* terminal of Relay
3. *NO* terminal of Relay $\longrightarrow$ Second terminal of Irrigation valve

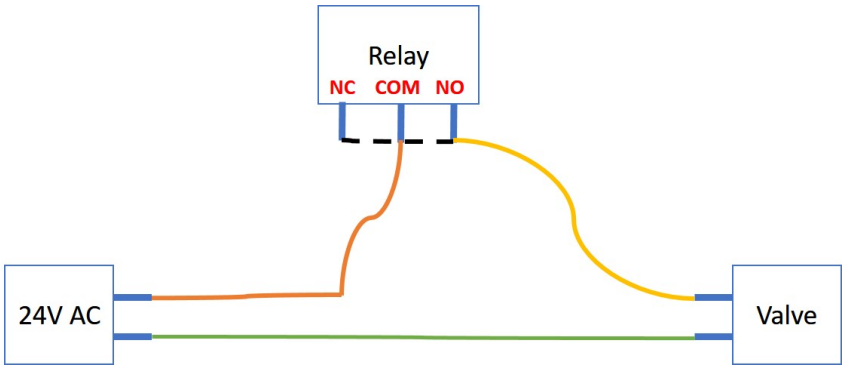


Figure 19: Relay-Valve connections

5 SERVER

5.1 Hardware Components

The following are the components needed for building a working server.

1. Intel NUC: https://goo.gl/3pFUJx
2. Ethernet Cable
3. Mini-HDMI to HDMI Cable
4. Monitor

In the above list, Ethernet cable, Mini-HDMI to HDMI Cable and monitor are already available in the lab which can be used. For building a new server, an Intel NUC will be needed else we already have one available.

5.2 Configuring and Installing Dependencies

5.2.1 Installing Ubuntu 16.04

1. Please download the following OS file using the Weblink:

   http://releases.ubuntu.com/16.04.3/ubuntu-16.04.3-desktop-amd6
   iso?_ga=2.263195717.836685332.1503297678-1846629868.
   1503297678

   The shortened version of the above URL. https://goo.gl/fgyyYQ

2. Please follow the instructions to install the Ubuntu Operating System in the Intel NUC: https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop

5.2.2 Installing Apache Server

The Apache web server is among the most popular web servers in the world. It's well-documented, and has been in wide use for much of the history of the web, which makes it a great default choice for hosting a website.

We can install Apache easily using Ubuntu's package manager, apt. A package manager allows us to install most software pain-free from a repository maintained by Ubuntu. You can learn more about how to use *apt* here.

For our purposes, we can get started by typing these commands:

```
sudo apt-get update
sudo apt-get install apache2
```

Since we are using a *sudo* command, these operations get executed with root privileges. It will ask you for your regular user's password to verify your intentions.

Once you've entered your password, *apt* will tell you which packages it plans to install and how much extra disk space they'll take up. Press *Y* and hit Enter to continue, and the installation will proceed.

5.2.3 Installing PHP

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display.

We can once again leverage the apt system to install our components. We're going to include some helper packages as well, so that PHP code can run under the Apache server and talk to our MySQL database:

```
sudo apt-get install php libapache2-mod-
   php php-mcrypt php-mysql
```

This should install PHP without any problems. We'll test this in a moment.

In most cases, we'll want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called index.html. We want to tell our web server to prefer PHP files, so we'll make Apache look for an index.php file first.

To do this, type this command to open the dir.conf file in a text editor with root privileges:

```
sudo nano /etc/apache2/mods-enabled/dir.
   conf
```

It will look like this:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
DirectoryIndex index.html index.cgi
   index.pl index.php index.xhtml index.
   htm
```

```
</IfModule>
```

We want to move the PHP index file highlighted above to the first position after the DirectoryIndex specification, like this:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
DirectoryIndex index.php index.html
   index.cgi index.pl index.xhtml index.
   htm
</IfModule>
```

When you are finished, save and close the file by pressing Ctrl-X. You'll have to confirm the save by typing Y and then hit Enter to confirm the file save location.

After this, we need to restart the Apache web server in order for our changes to be recognized. You can do this by typing this:

```
sudo systemctl restart apache2
```

We can also check on the status of the apache2 service using systemctl:

```
sudo systemctl status apache2
```

Sample Output

```
apache2.service - LSB: Apache2 web
   server
Loaded: loaded (/etc/init.d/apache2; bad
   ; vendor preset: enabled)
Drop-In: /lib/systemd/system/apache2.
   service.d
apache2-systemd.conf
Active: active (running) since Wed
   2016-04-13 14:28:43 EDT; 45s ago
Docs: man:systemd-sysv-generator(8)
Process: 13581 ExecStop=/etc/init.d/
   apache2 stop (code=exited, status=0/
   SUCCESS)
Process: 13605 ExecStart=/etc/init.d/
   apache2 start (code=exited, status=0/
   SUCCESS)
Tasks: 6 (limit: 512)
CGroup: /system.slice/apache2.service
13623 /usr/sbin/apache2 -k start
```

```
13626 /usr/sbin/apache2 -k start
13627 /usr/sbin/apache2 -k start
13628 /usr/sbin/apache2 -k start
13629 /usr/sbin/apache2 -k start
13630 /usr/sbin/apache2 -k start

Apr 13 14:28:42 ubuntu-16-lamp systemd
   [1]: Stopped LSB: Apache2 web server.
Apr 13 14:28:42 ubuntu-16-lamp systemd
   [1]: Starting LSB: Apache2 web server
   ...
Apr 13 14:28:42 ubuntu-16-lamp apache2
   [13605]:  * Starting Apache httpd web
    server apache2
Apr 13 14:28:42 ubuntu-16-lamp apache2
   [13605]: AH00558: apache2: Could not
   reliably determine the server's fully
    qualified domain name, using
   127.0.1.1. Set the 'ServerNam
Apr 13 14:28:43 ubuntu-16-lamp apache2
   [13605]:  *
Apr 13 14:28:43 ubuntu-16-lamp systemd
   [1]: Started LSB: Apache2 web server.
Install PHP Modules
```

To enhance the functionality of PHP, we can optionally install some additional modules.

To see the available options for PHP modules and libraries, you can pipe the results of apt-cache search into less, a pager which lets you scroll through the output of other commands:

```
apt-cache search php- | less
```

Use the arrow keys to scroll up and down, and q to quit.

The results are all optional components that you can install. It will give you a short description for each:

```
libnet-libidn-perl - Perl bindings for
   GNU Libidn
php-all-dev - package depending on all
   supported PHP development packages
php-cgi - server-side, HTML-embedded
   scripting language (CGI binary) (
   default)
php-cli - command-line interpreter for
   the PHP scripting language (default)
```

```
php-common - Common files for PHP
   packages
php-curl - CURL module for PHP [default]
php-dev - Files for PHP module
   development (default)
php-gd - GD module for PHP [default]
php-gmp - GMP module for PHP [default]
```

To get more information about what each module does, you can either search the internet, or you can look at the long description of the package by typing:

```
apt-cache show package_name
```

There will be a lot of output, with one field called Description-en which will have a longer explanation of the functionality that the module provides.

For example, to find out what the php-cli module does, we could type this:

```
apt-cache show php-cli
```

Along with a large amount of other information, you'll find something that looks like this:

Output

```
Description-en: command-line interpreter
    for the PHP scripting language (
   default)
This package provides the /usr/bin/php
   command interpreter, useful for
testing PHP scripts from a shell or
   performing general shell scripting
   tasks.
.
PHP (recursive acronym for PHP:
   Hypertext Preprocessor) is a widely-
   used
open source general-purpose scripting
   language that is especially suited
for web development and can be embedded
   into HTML.
.
This package is a dependency package,
   which depends on Debian's default
```

```
PHP version (currently 7.0).
```

If, after researching, you decide you would like to install a package, you can do so by using the apt-get install command like we have been doing for our other software.

If we decided that *php-cli* is something that we need, we could type:

```
sudo apt-get install php-cli
```

If you want to install more than one module, you can do that by listing each one, separated by a space, following the apt-get install command, like this:

```
sudo apt-get install package1 package2
```

5.2.4 Test PHP Processing on your Web Server

In order to test that our system is configured properly for PHP, we can create a very basic PHP script.

We will call this script *info.php*. In order for Apache to find the file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".

In Ubuntu 16.04, this directory is located at */var/www/html/*. We can create the file at that location by typing:

```
sudo nano /var/www/html/info.php
```

This will open a blank file. We want to put the following text, which is valid PHP code, inside the file:

```
                    info.php
<?php
phpinfo();
?>
```

When you are finished, save and close the file.

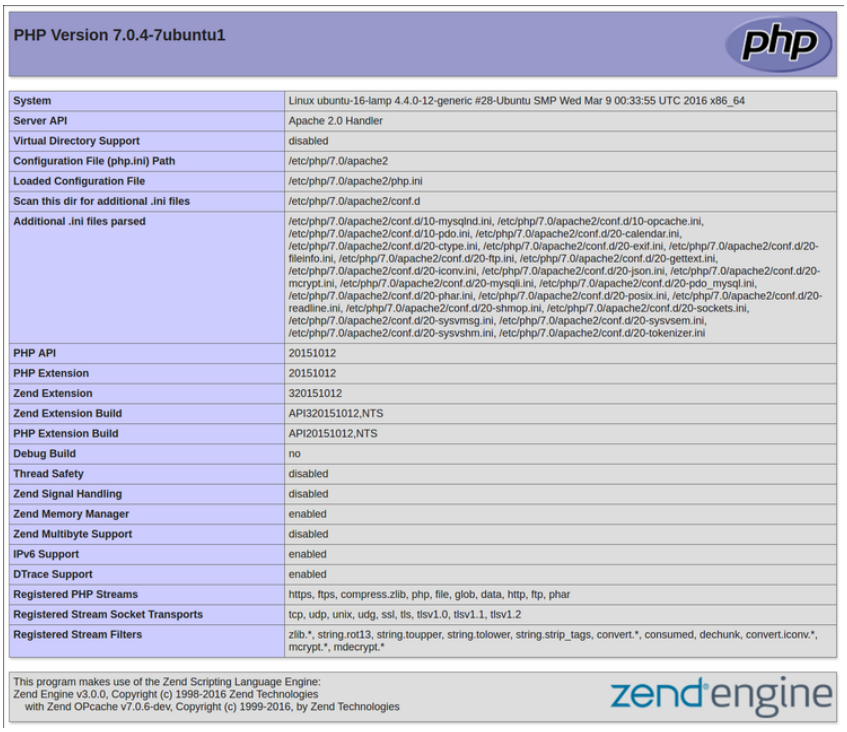
Now we can test whether our web server can correctly display content generated by a PHP script. To try this out, we just have to visit this page in our web browser. You'll need your server's public IP address again.

The address you want to visit will be:

```
http://your_server_IP_address/info.php
```

The page that you come to should look something like this:

Ubuntu 16.04 default PHP info



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

Figure 20: PHP default landing page

This page basically gives you information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If this was successful, then your PHP is working as expected.

You probably want to remove this file after this test because it could actually give information about your server to unauthorized users. To do this, you can type this:

```
sudo rm /var/www/html/info.php
```

You can always recreate this page if you need to access the information again later.

5.2.5 Installing Databases (SQLite)

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is the most used database engine in the world.

To install SQLite on the server follow the instructions described below:

1. Open the terminal and type the following command.

```
sudo apt-get install php5-sqlite
```

5.2.6 IP configuration

Static IP Address Assignment

To configure your system to use a static IP address assignment, add the static method to the *inet* address family statement for the appropriate interface in the file **/etc/network/interfaces**. The example below assumes you are configuring your first Ethernet interface identified as *eth0*. Change the address, netmask, and gateway values to meet the requirements of your network.

```
\# interfaces(5) file used by ifup(8)
   and ifdown(8)
auto lo eth0
iface lo inet loopback
iface eth0 inet static
address 165.91.212.81
netmask 255.255.248.0
network 165.91.208.0
gateway 165.91.208.1
broadcast 165.91.215.255
dns-nameservers 128.194.254.1
   128.194.254.2 128.194.254.3
```

Configure the DNS servers has been included in the above file.

Restart networking

```
sudo /etc/init.d/networking restart
```

Or

```
systemctl restart ifup@eth0
```

Firewall

Used working file **rules.v4**: Located at **/etc/iptables/rules.v4**

```
# Generated by iptables-save v1.4.21 on
   Fri Sep 16 00:05:22 2016
*filter
```

```
:INPUT ACCEPT [3009:293827]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2427:459561]
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j
   ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j
   ACCEPT
-A INPUT -m conntrack --ctstate RELATED,
   ESTABLISHED -j ACCEPT
COMMIT
# Completed on Fri Sep 16 00:05:22 2016
```

Installing the Persistent Firewall Service To get started, you will need to install the iptables-persistent package if you have not done so already. This will allow us to save our rule sets and have them automatically applied at boot:

```
sudo apt-get update
sudo apt-get install iptables-persistent
```

To implement our firewall policy and framework, we will be editing the **/etc/iptables/rules.v4** and **/etc/iptables/rules.v6** files. Open the rules.v4 file in your text editor with *sudo* privileges:

To view the IPtables:

iptables uses three different chains: input, forward, and output.

Input – This chain is used to control the behavior for incoming connections. For example, if a user attempts to SSH into your PC/server, iptables will attempt to match the IP address and port to a rule in the input chain.

Forward – This chain is used for incoming connections that aren't actually being delivered locally. Think of a router – data is always being sent to it but rarely actually destined for the router itself; the data is just forwarded to its target. Unless you're doing some kind of routing, NATing, or something else on your system that requires forwarding, you won't even use this chain.

list the current rules that are configured for iptables

```
iptables -L
```

We can see the output in a format that reflects the commands necessary to enable each rule and policy by instead using the -S flag:

```
sudo iptables -S
```

Reset your Firewall We will start by resetting our firewall rules so that we can see how policies can be built from the command line. You can flush all of your rules by typing:

```
sudo service iptables-persistent flush
```

OR If you do have rules in place and wish to scrap them and start over, you can flush the current rules by typing:

```
sudo iptables -F
```

Writing the Rule we need is this:

```
sudo iptables -A INPUT -m conntrack --
   ctstate ESTABLISHED,RELATED -j ACCEPT
```

Delete Rule by Specification One of the ways to delete iptables rules is by rule specification. To do so, you can run the iptables command with the -D option followed by the rule specification. If you want to delete rules using this method, you can use the output of the rules list, *iptables -S*, for some help.

For example, if you want to delete the rule that drops invalid incoming packets *(-A INPUT -m conntrack –ctstate INVALID -j DROP)*, you could run this command:

```
sudo iptables -D INPUT -m conntrack --
   ctstate INVALID -j DROP
```

Note that the -A option, which is used to indicate the rule position at creation time, should be excluded here.

Delete Rule by Chain and Number

The other way to delete iptables rules is by its chain and line number. To determine a rule's line number, list the rules in the table format and add the –line-numbers option:

```
sudo iptables -L --line-numbers
[secondary_output Example Output: Rules
   with Line Numbers]
Chain INPUT (policy DROP)
num  target     prot opt source
                 destination
```

```
1    ACCEPT     all  --  anywhere
                anywhere
   ctstate RELATED,ESTABLISHED
2    ACCEPT     all  --  anywhere
                anywhere
3    DROP       all  --  anywhere
                anywhere
   ctstate INVALID
4    UDP        udp  --  anywhere
                anywhere
   ctstate NEW
5    TCP        tcp  --  anywhere
                anywhere             tcp
   flags:FIN,SYN,RST,ACK/SYN ctstate NEW
6    ICMP       icmp --  anywhere
                anywhere
   ctstate NEW
7    REJECT     udp  --  anywhere
                anywhere
   reject-with icmp-port-unreachable
8    REJECT     tcp  --  anywhere
                anywhere
   reject-with tcp-reset
9    REJECT     all  --  anywhere
                anywhere
   reject-with icmp-proto-unreachable
10   ACCEPT     tcp  --  anywhere
                anywhere             tcp
   dpt:ssh ctstate NEW,ESTABLISHED
...
```

This adds the line number to each rule row, indicated by the num header.

Once you know which rule you want to delete, note the chain and line number of the rule. Then run the *iptables -D* command followed by the chain and rule number.

For example, if we want to delete the input rule that drops invalid packets, we can see that it's rule 3 of the INPUT chain. So we should run this command:

```
sudo iptables -D INPUT 3
```

Now that you know how to delete individual firewall rules, let's go over how you can flush chains of rules.

Saving Updates If you ever update your firewall and want to preserve the changes, you must save your iptables rules for them to be persistent. Save your firewall rules with this command:

```
sudo invoke-rc.d iptables-
    persistent save
```

Reference:

Following references were used for setting up the firewall.

1. https://goo.gl/hqnyC6
2. https://goo.gl/vnJGFw
3. https://goo.gl/bNvblg
4. https://goo.gl/o6cxPI

5.2.7 Installing Python

Python 3 is installed by default on modern versions of Ubuntu, so you should already have it installed:

```
python3 -V
```

5.2.8 Installing Django

Django is a web application framework written in python that follows the MVC (Model-View-Controller) architecture, it is available for free and released under an open source license. It is fast and designed to help developers get their application online as quickly as possible. Django helps developers to avoid many common security mistakes like SQL Injection, XSS, CSRF and clickjacking. Django is maintained by the Django Software Foundation and used by many big technology companies, government, and other organizations. Some large websites like Pinterest, Mozilla, Instagram, Discuss, The Washington Post etc. are developed with Django.

Please follow the tutorial on how to install Django on Ubuntu. https://goo.gl/qpkjAH

5.3 Deploying Web Interface

1. Download the server logic and interface files from the dropbox link: Willbeupdated.
2. Extract the files using the command.

```
mkdir temp_for_zip_extract
unzip /path/to/file.zip -d
   temp_for_zip_extract
```

3. After extracting the files, copy them to the following folder.

```
sudo cp -R temp_for_zip_extract /var/
   www/html
```

4. Now change the permission of the files using the command

```
cd /var/www/html
sudo  chmod -R 755 *
```

5.4 Explanation of Main functionality of the server

This section describes the additional features that are used for building and setting up the server.

5.4.1 Security Feature

To add or delete authorized users, please follow the steps shown:

1. Execute the script in a terminal window using the command to add user.

```
python /Desktop/Server/login/script.py
```

2. When prompted fill in all the details of the user.

5.4.2 Configuring Cron-Job

We use cron to setup and execute the script that downloads weather data. Please follow the general tutorial to setup the cron job `https://www.taniarascia.com/setting-up-a-basic-cron-job-in-li`

The only difference between the tutorial and ours is that the script files are different. So please use this for setting up a cron job for our project.

```
*/15 * * * *    /usr/bin/python Desktop/
   WeatherGraber/WeatherGraber/
   weatherpull_scheduler.py
```

5.4.3 Downloading Weather Data

One of the main goals of this study is to build a smart operational support for LIRMS. To build such a smart support system, environmental conditions are needed to ensure proper operation. Therefore, continuous monitoring and storing of the weather parameters is an important step in achieving this goal. To continuously monitor and store the weather parameters, a python script has been implemented on the server. The script downloads the current and forecast weather data using the Open Weather Map API (`http://www.openweathermap.com/api`).

The weather data is collected every fifteen minutes. Table 1 describe the parameters provided by the API. The script downloads the weather data using the API in the form of a JSON object, which is parsed and stored in an SQLite database named *weather.db*.

Table 1: Current weather parameters downloaded and stored in the weather database.

Parameter Name	Parameter Description	Units
C_Temp	Current Temperature.	Kelvin
C_Humidity	Current Humidity	%
C_Pressure	Current Atmospheric Pressure	hPa
C_Temp_min	Minimum Temperature at the moment	Kelvin
C_Temp_max	Maximum Temperature at the moment	Kelvin
C_Wind_speed	Current Wind Speed	meter/sec
C_Wind_deg	Current Wind Direction	degrees
C_Clouds	Current Cloudiness	%
C_Rain	Rain volume for the last 3 hours	inches
C_Snow	Snow volume for the last 3 hours	inches

Starting from October 2016, more than 11K weather readings have been downloaded and stored in the database. The weather data can be displayed using the URL (`http://lirms.tamu.edu/weatherplots`).

To Download the weather data, please run the following code in the terminal.

```
cd Desktop/WeatherGraber/WeatherGraber
```

The above steps can be shown in the figure 21. To make sure if the script file exists in the folder please type *ls* command. The figure 21 shows the files listed in the folder.

We need to run the *weatherpull_scheduler.py* script. The below command is used to run the script.

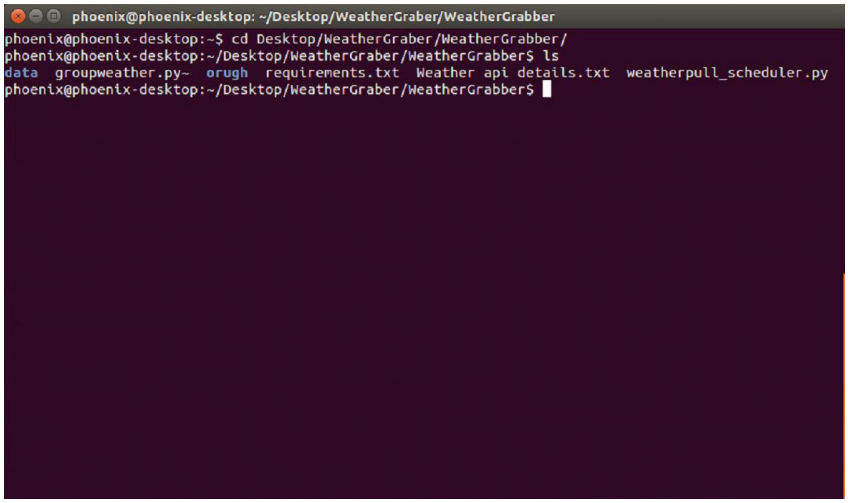


Figure 21: Weather grabbing script

```
python weatherpull_scheduler.py
```

The figure 22 shows the weather data collected by the script.

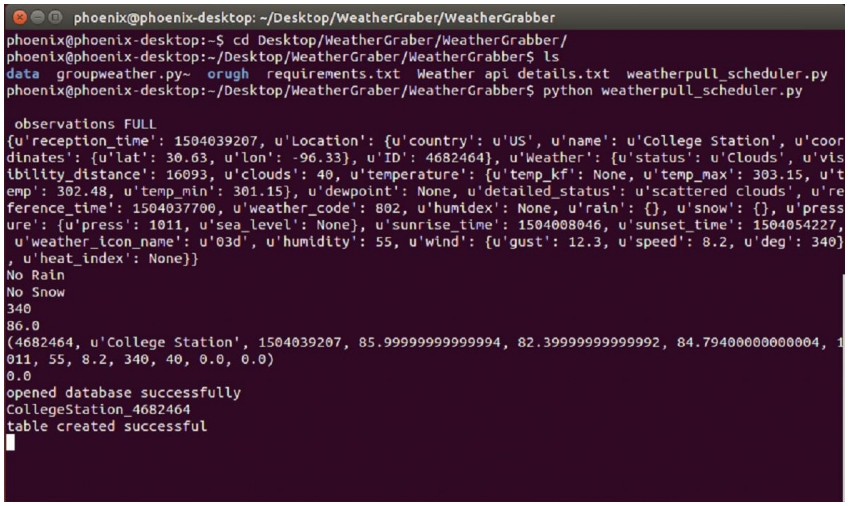


Figure 22: Weather data collected by the script in JSON format

The downloaded data is then stored in the SQLite database stores in the root folder of the server which can be accessed using the web-interface available at (`http://lirms.tamu.edu/weatherplots`)

5.4.4 Visualizing

For the visualizing feature, Google charts API is used. Google charts API is a Javascript based client-side dynamic tool.

5.4.5 Programming Irrigation Cycle

1. Login to the `lirms.tamu.edu` website. Credentials: (username: lastname; password: lirms@lastname)
2. Click on "Program" weblink in the navigation menu. or follow this link here: `http://lirms.tamu.edu/program.php`.
3. Select the controller you want to program. Ex: select "Dallas unit" for programming LIRMS installed at Dallas. figure 23

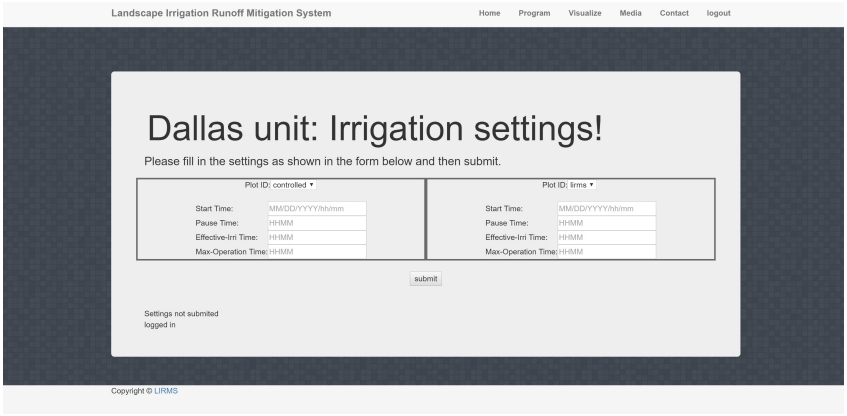


Figure 23: Dallas unit irrigation settings form

4. Since only one plot is set to irrigate at a given time, fill in the irrigation settings accordingly. Starting with the "Controlled plot" and then go to the "LIRMS plot". So the "starting time" for each should be separated with at least "Effective-irrigation time". You have to fill in all the sections before you submit as shown in figure 24.

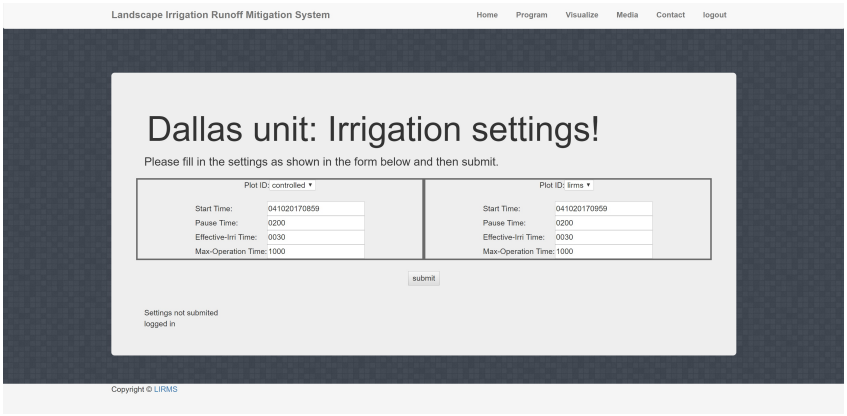


Figure 24: After filling the form

5. After filling in all the irrigation settings accordingly, hit Submit. You will receive a message at the bottom of the page

saying: "Will be Downloaded to the lirms in few seconds" as shown in the figure 25.

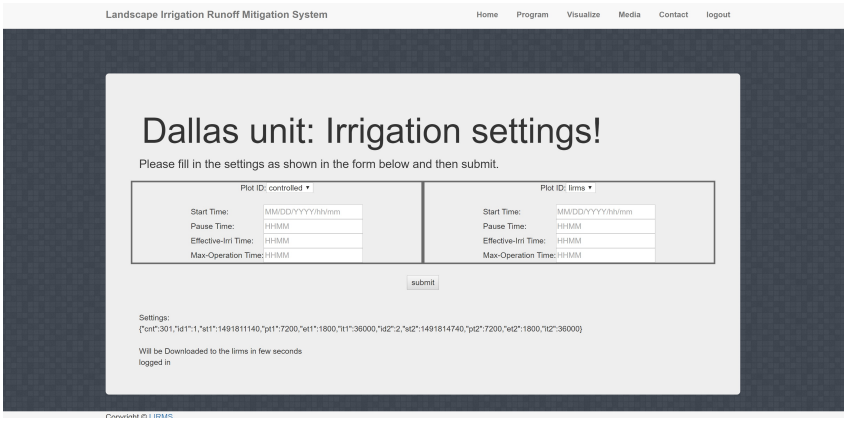


Figure 25: After submission

6. **Following the above steps, we can program an irrigation cycle.**

5.4.6 Visualizing Irrigation Cycle

A web-based user interface (UI) is designed for visualizing and analyzing an irrigation cycle as shown in the figure 26. The UI can be accessed using the URL (`http://lirms.tamu.edu/dloggervisual`). The visualizing interface is powered using Google chart API (`https://developers.google.com/chart/`). Google chart API is an interactive web service that creates graphical charts from data.

A php logic is implemented to create graphical charts on-the-fly using the corresponding or relevant data. A date based visualizing UI has been developed. The graphical charts can be created with the click of a button. In the backend, the python script retrieves the necessary data for the selected date and creates graphical charts using the Google chart API.

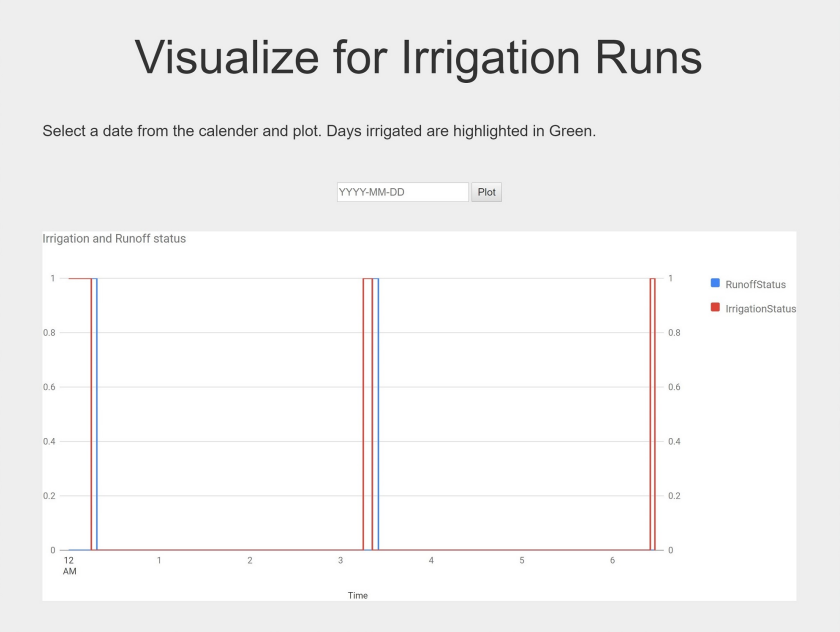


Figure 26: Web-based User Interface for visualizing irrigation activity. User can plot the irrigation activity for each irrigation cycle.