SubsySTEM

3

SUBSYSTEMS

Internet of Things (IoT)

Table of Contents

Introduction

The internet has changed our lives in many ways. Today, a world of information is available right at our fingertips. With a quick click of a button, you can learn about foreign culture, listen to music, watch your favorite TV show, or spend hours looking at videos of funny cats.

The internet has also allowed us to connect to the world in ways we weren’t able to before. With our cell phones we can now purchase merchandise, unlock our car doors, and view the video surveillance of our homes, all while on the go.

Our devices are connected to the internet as well. We have systems that would have been considered pure science fiction a couple of decades ago. We can speak into the air and our internet connected devices can play music, turn on lights, order products, and let us know what the weather is like.

These devices and many more like them are all a part of the **Internet of Things** (normally abbreviated **IoT**). This phase refers to the myriad of people, appliances, vehicles, sensors, cameras, and other devices that are connected to the internet and send and receive information. It has spawned an immense industry and has set in motion a path for future living.

In this Subsystem Module, you will create your own IoT device, program it to run on your local network, and use it to monitor your home while you are out. You will become a part of the Internet of Things.

Objectives

When completed with this Subsystem Module, the student will:

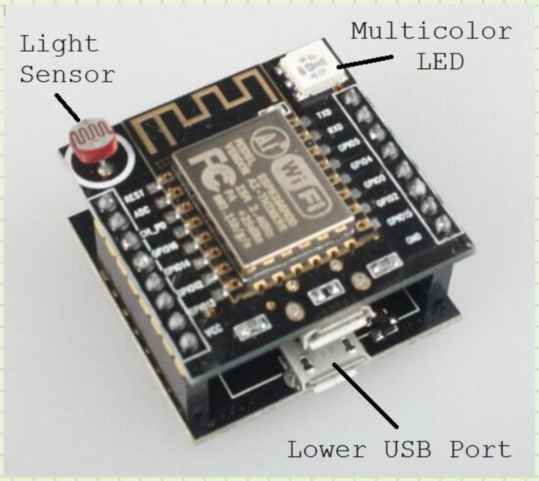
1) be able to explain what the Internet of Things is and how it is used.
2) describe what is meant by "The Cloud" with respect to computers and the internet.
3) program a small computer with the ability to connect to the internet.
4) link to a cloud server to communicate remotely with your IoT computer.
5) interact with different sensors on your computer to be able to monitor those sensor readings remotely.

Tools

To accomplish these objectives, you have a few tools at your disposal.

First, you have this curriculum guide. It will walk you step by step through the entire curriculum that includes the text, system setup, and programming examples.

You also have **FRED** (your FRiendly Educational Device). He is a tiny computer with USB interface and built-in Wi-Fi. He gives you all you need to connect to the IoT. He includes a lower USB port for power and programming (the upper port can be used for power only). There is a light sensor for input and a multicolor (Red/Green/Blue) LED (light emitting diode) for output. There are also 3 pushbutton switches. On the bottom board there are two labeled RST and FLASH and are used for programming. The switch on the top board is for user input.



Section 1

The Internet

Our discussion of the Internet of Things must surely start with the Internet itself.

The **Internet** (commonly, the Net) is a global system of interconnected computer networks that use agreed upon standards to link worldwide network systems together. It is not one system, but a combination of countless computer networks that have all been tied using a common framework for operations. It grew much like our wired telephone system. Telephones in a particular building were linked to an operator to connect different rooms together. Then buildings were linked together so people from different buildings could communicate. Then, companies installed wires to allow houses to be connected until the whole town could talk to one another. Then, cables brought the ability to link towns together. As we figured out how to automate the control of how calls got connected, even more phones could be hooked together and calls could be completed faster. Soon, the whole country was hooked into this phone system.

Now replace these phones with computers. And connect all of these computers with not just wires but wireless radios and fiber optic cables. Then, connect countries together on the same network so we can all share the information present on this system. You now have our modern internet.

The internet was not only possible because of the computer connections. It is also possible because these connections all speak the same language. Early in the history of the internet, groups decided on standards for this data transfer which led to the structure we have today.

Let’s look how our old telephone system is different than today’s internet.

The original telephone system in the US consisted of individual phones, wires, and connecting systems. When you placed a call, switching networks made of relays would use the number you dialed on the phone to set up a specific path to another phone on the network. This was a dedicated line of communication. Two phones were connected through miles of wire to connect each other together. You could then talk to someone on the other end of the line and they could talk back. This is amazingly inefficient because when you weren't actually talking, there was no data being transferred on these lines. And if anything happened to your connection, the call was just dropped. As digital systems became available, we were able to design more reliable ways to use these communication lines.

Enter todays internet. Information is sent over the internet in **packets**. Let's say you want to send an email to your friend. You press the send button and you set in motion a complicated series of events. Your email is broken into packets (normally 1500 bytes of data, but could be up to 64 kbytes). A header and footer (commonly called a "**wrapper**") is then added to each packet with information about the type of content in the packet, how it fits together with other packets, the origin of the packet, and the ultimate destination.

The internet then routes these packets to the destination by jumping from site to site, always picking a site closer to the final destination. When all the packets arrive, the local site uses the information in the wrapper to reassemble the original content and route it to the correct program. Poof, your email shows up in your friend's inbox.

Some of the advantages of this system is that the packets can show up to the destination by any means. Some may go through wires, others over radio, and still others over fiber optics which use light to transfer information. The packets of your email don't all have to travel over the same route either.

And at the destination, if a packet is missing, the destination system can ask that originating system to retransmit the packet again. This is a very reliable system with some built in error checking. It may seem like this would take a long time to accomplish, but it only takes a couple of milliseconds (thousandths of a second) to complete. If one path of information flow gets crowded, the packets can be sent over alternate paths. This keeps lots of information flowing constantly and adds to the reliability of the network.

The Cloud

You may have heard this term used before. The “Cloud” isn’t actually a thing as it is a concept (and is frequently referred to as Cloud Computing). Normally, this refers to performing some kind of computing with resources that are located somewhere else on the internet. For instance, many companies offer storage for your pictures and music. This will take the form of large memory banks (hard drives) located somewhere on the internet. This company can control access so that only you can view the files in your account. They are responsible to maintain your information and their connection to the internet. This frees up your device memory for apps and other things. Many businesses are moving their collaborative software to the Cloud. This will allow their employees to have access to the applications they need (word processing, spreadsheets, picture and video editing software) as well as the files they are working on no matter where they are. As the internet becomes faster and more reliable, you can expect this type of computing to increase and the Cloud to be a part of everyday life.

With this foundation behind us, we can now discuss the concept of the **Internet of Things**.

Section

2

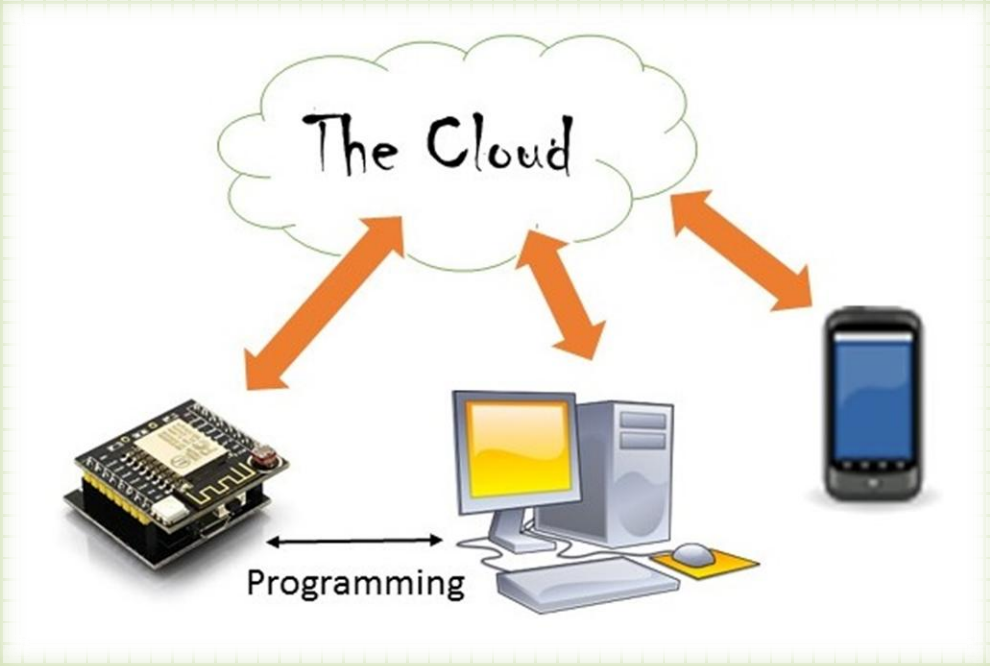
The Internet of Things

The **Internet of Things** (commonly, **IoT**) is the collection of devices that are tied into the internet that collect and exchange data. It is a loose concept since we wouldn't consider your internet connected computer necessarily an IoT device. It is meant more to express a device that is collecting information on something and that device is accessible from the internet. Here are some examples to help clarify:

1) Biochip transponders on farm animals. These track the animal's vital signs and location.
2) An automobile where the airbag sensors are monitored remotely.
3) Thermostats that allow you to set the temperature in your home remotely.
4) A button on your washing machine that allows you to reorder laundry detergent by just pressing it.
5) A remote weather monitoring station that broadcasts its data continuously on an internet site.
6) Your electric meter that reports energy usage and power outages automatically to the power company.

There are many more examples but hopefully you get the idea. This basic idea can also be abstracted to more complex ideas like smart buildings, smart cities, and smart grids. Think of how nice it would be if traffic lights knew exactly where all the cars were at any given time. If there weren't a lot of cars on the road, it is feasible that the traffic lights could sequence themselves so you would not have to stop at any of them. That would save a ton of gas, time, and frustration. There are also real concerns. If you decide to hook your house locks into the internet so you can open your home with your cell phone, you run the risk of someone figuring out how to hack into that system and get access to your home. This is the constant battle between security and convenience.

The way we are going to become part of the IoT is by connecting FRED to the internet and programming it to respond to requests.



The above diagram shows how we will connect everything together to become part of the Cloud.

First, we will need to program FRED to get on to your local wireless internet connection. You will do this by loading software on your computer that is used to program devices like FRED. It is a wildly popular software package called Arduino (www.arduino.cc). Arduino (Ar-dween-oh) is the trade name for a group of boards, add on modules, and programming software for the hobby community. It is used to program robots, science fair projects, home automation, you name it. You won't actually be doing any detailed programming in this module, but the Arduino software is free, easy to use, and has all of the needed code to get our project going.

That is going to allow us to program the Wi-Fi information into FRED. But FRED needs more than that to work. For this, we will load him with Blynk libraries that work with the Blynk Cloud Server. Blynk is a company that offers individuals and companies easy access to IoT devices. Their software allows us to have FRED communicate with the cloud server by receiving request and sending data. After downloading this code, FRED will no longer need to be connected to our computer to function on line. At this point, FRED is online and part of the IoT.

The last step is sending requests to the Blynk server to get information from FRED. Luckily, Blynk offers an iOS and Android App to do just that. When you download this app to your phone or tablet, you will be able to configure it to talk to FRED. And the great part is that this communication happens via the Blynk cloud server so you will able to access the information in FRED from anywhere in the world that you have internet access. By dealing with the cloud server, we do not need to fumble with the awkward issue of changing our home internet router settings to allow outside sources from getting information from our internal network. Most avid video gamers know how to do this because it is usually required to play a lot of the online multi-player games they love. But even most of them will tell you it is a hassle. Because our phone and computer both connect outside of our local network (through the Blynk server), we do not need to mess with this.

Don’t worry if this seems like a lot of technical stuff. We will slowly work through it all step by step. The goal here is to understand the concept of the connection we are making. We will hook FRED into the internet and register him with the Blynk server. Then we will download an app to connect us to the Blynk server. Then we will communicate to FRED through the Blynk server. Blynk has done all the hard work for us. Now we just need to move on to step 1.

Let’s load up Arduino and get it configured.

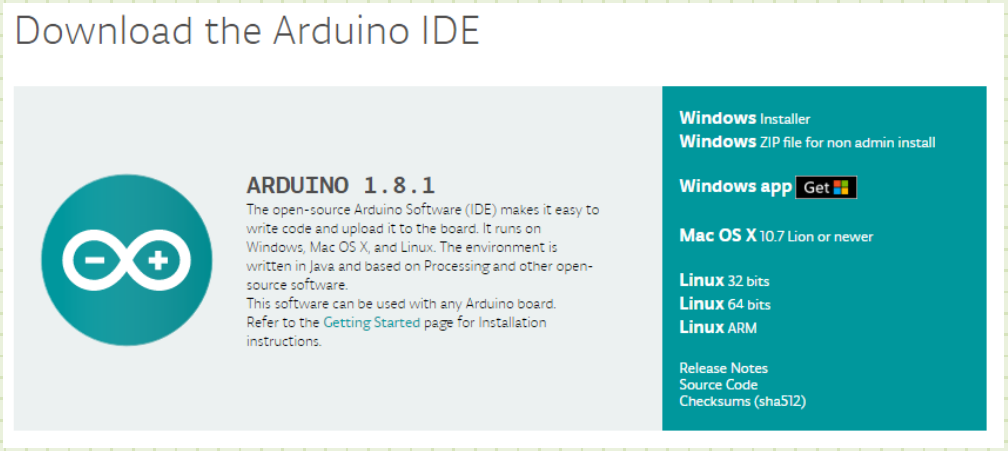
Section 3

Arduino Software Load

We will start our setup with loading the Arduino software and making sure it is configured properly. If you already have a current copy of Arduino on your computer, you can skip down to the customization steps below.

Visit www.arduino.cc and download the latest version of the software for your particular system. Go to the main site and click on the menu item "Software" on the top navigation banner.

Scroll down until you find the link to download the IDE (integrated development environment).



Download the Arduino IDE

ARDUINO 1.8.1

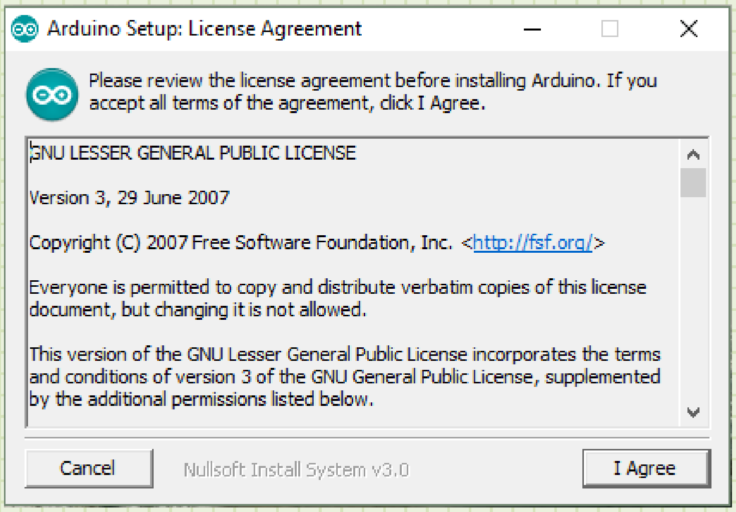
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install
Windows app Get
Mac OS X 10.7 Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM
Release Notes
Source Code
Checksums (sha512)

On the right, is a list of the different operating systems. Click on the one that corresponds to the one you use. Opt for the full version and not the app if it is offered.

Download this file to a place on your computer where you can find it. Windows will normally put downloads in the "Downloads" folder.

Now, run that installation program. You will be greeted with a user agreement.



Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

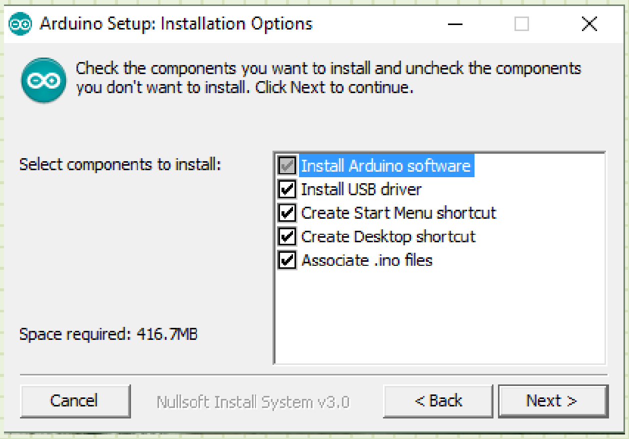
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

If you agree with the license, click "I agree" to move on to an "Installation Options" dialog.



Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

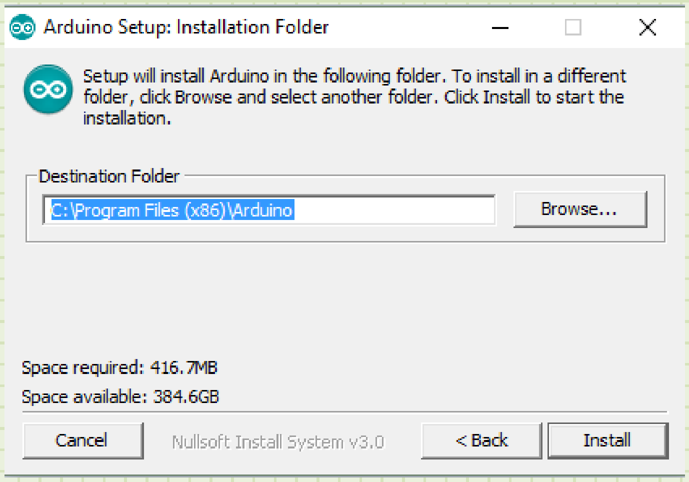
Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 416.7MB

The default values are good for our purposes so click "Next."

You will then be presented with a dialog that has you choose the installation location.


Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

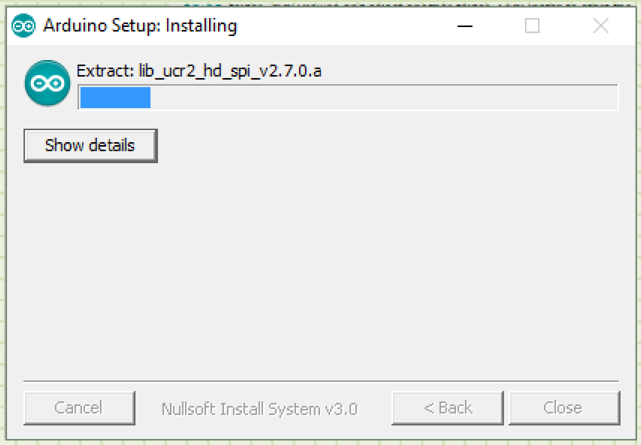
C:\Program Files (x86)\Arduino

Space required: 416.7MB

Space available: 384.6GB

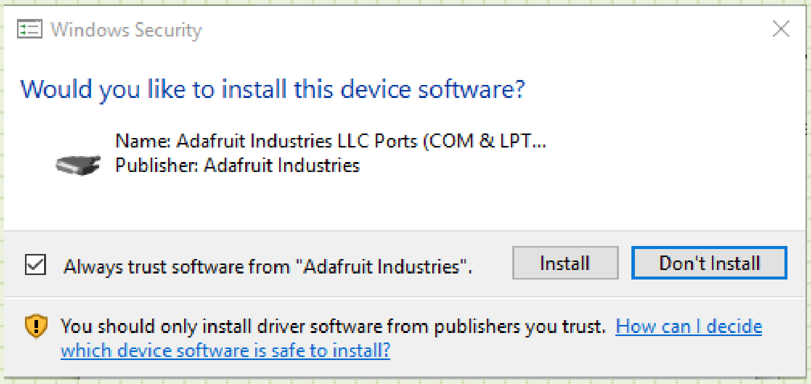
And again, the default value works great for us so just click "Install."

The installation software will start loading the program and all the support files.


Arduino Setup: Installing

Extract: lib_ucr2_hd_spi_v2.7.0.a

This installation will take a little bit to complete. At some time, you may be presented with driver installation warnings like the following:


Windows Security

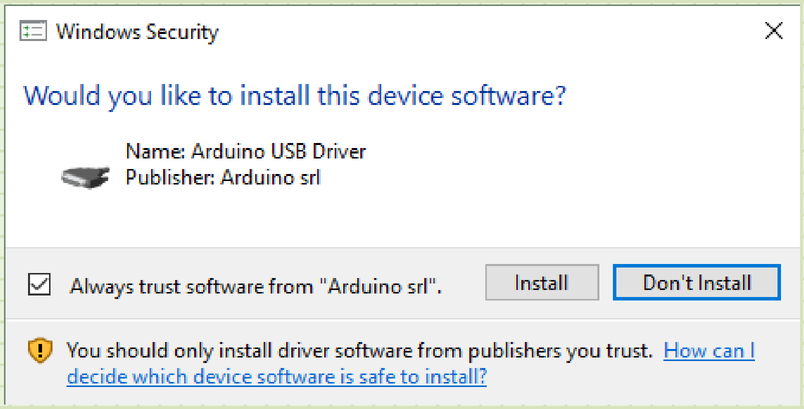
Would you like to install this device software?

Name: Adafruit Industries LLC Ports (COM & LPT...

Publisher: Adafruit Industries

☑ Always trust software from "Adafruit Industries".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?


Windows Security

Would you like to install this device software?

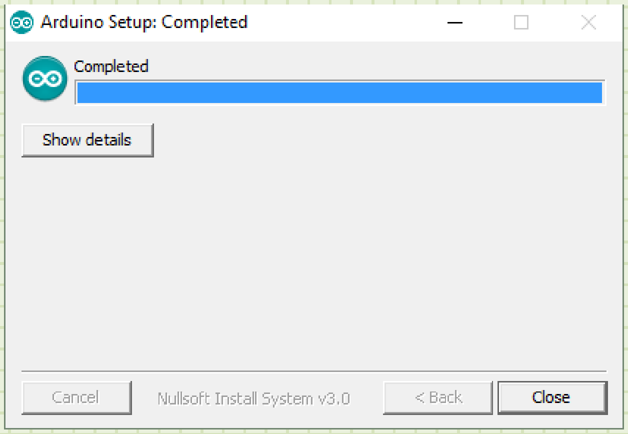
Name: Arduino USB Driver
Publisher: Arduino srl

☑ Always trust software from "Arduino srl".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

These are trusted, well know sites that develop software for the hobby community so click "Install" to install these drivers. These will allow us to communicate over the USB port with FRED.

When the installation is complete, you will be presented with a final dialog.

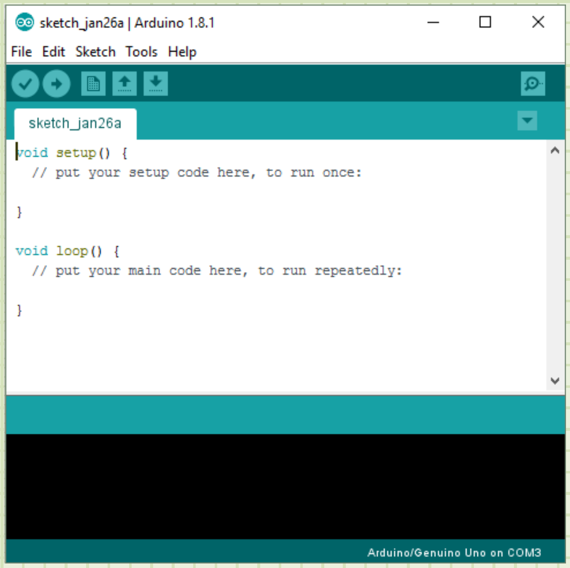

Arduino Setup: Completed

Completed

Click the "Close" button.

If all, went well, the Arduino software and drivers are loaded on your computer. Let's open up that software and finish setting it up.

Find the Arduino program on your computer (from the Start icon in Windows or the Launchpad on a Mac) and run it. You should see the following:

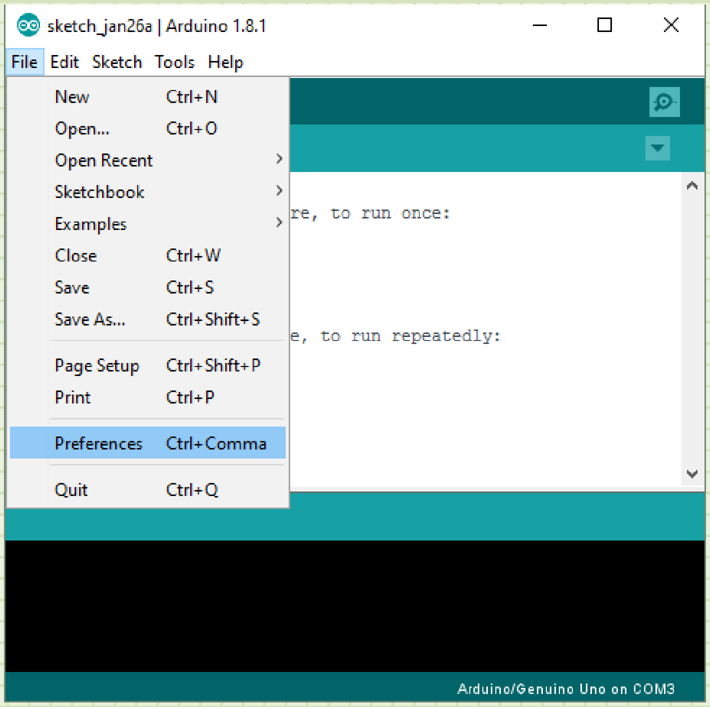

sketch_jan26a | Arduino 1.8.1
File Edit Sketch Tools Help
sketch_jan26a
Arduino/Genuino Uno on COM3

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

If you are presented with any other dialogs that show options for downloading other components, you can either accept them or cancel them. Your base software has everything we need.

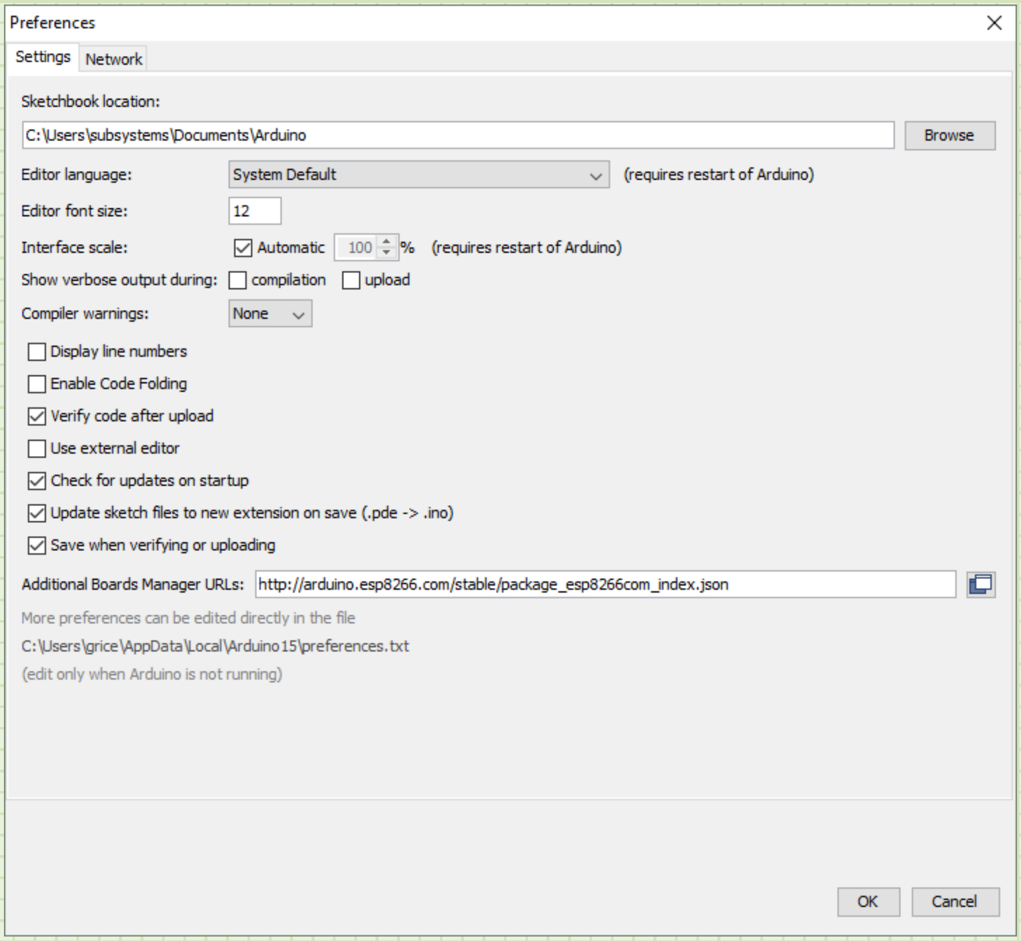
Now, we want to load the libraries to be able to talk to FRED. Arduino software has been updated to make this process much easier.

Select **File->Preferences.**



```
re, to run once:

e, to run repeatedly:
```

What the software allows you to do is enter the online address of stored libraries and it will automatically fetch them for us. It will also allow the software to determine when updates are available.



Preferences

Settings | Network

Sketchbook location:

C:\Users\subsystems\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

[ ] Display line numbers

[ ] Enable Code Folding

[✓] Verify code after upload

[ ] Use external editor

[✓] Check for updates on startup

[✓] Update sketch files to new extension on save (.pde -> .ino)

[✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

C:\Users\grice\AppData\Local\Arduino15\preferences.txt

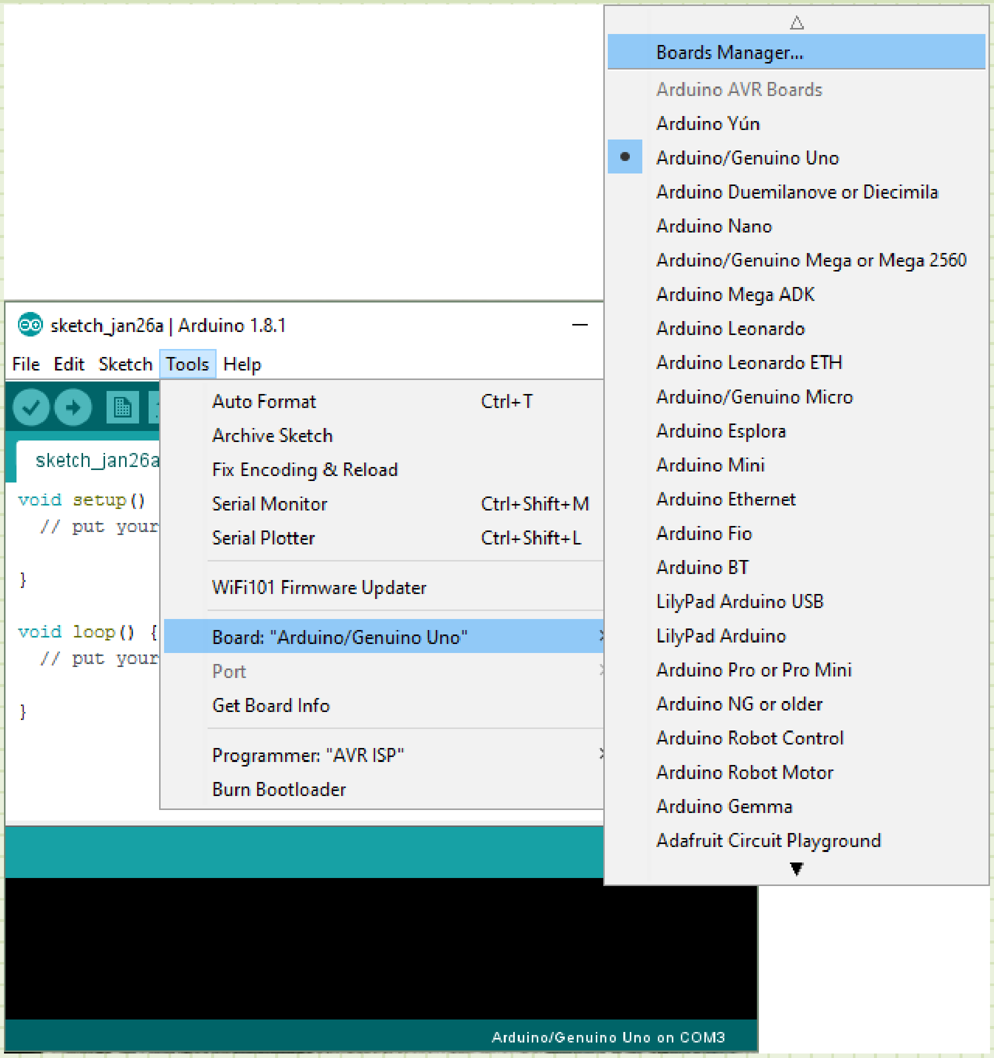
(edit only when Arduino is not running)

In this Preferences section, add the following line to the "Additional Boards Manager URL's" entry box:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

This is the repository for the files needed to talk to FRED. Click "OK."

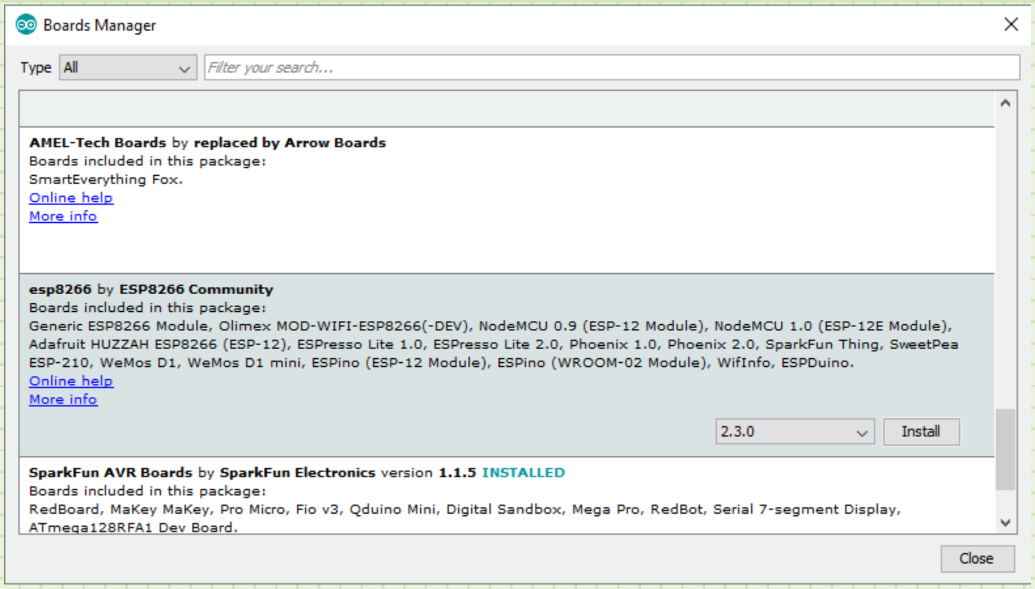
Now, we need to tell the Arduino software to go get these files from the repository.



```
void setup()
  // put your
}

void loop() {
  // put your
}
```

Select **Tools->Board:->Boards Manager…**

You will be presented with the libraries for various boards supported by the software. Scroll down until you see the esp8266 board entry. Click on this box. When you do, the option to install these boards will appear as shown below.



AMEL-Tech Boards by **replaced by Arrow Boards**
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info
2.3.0

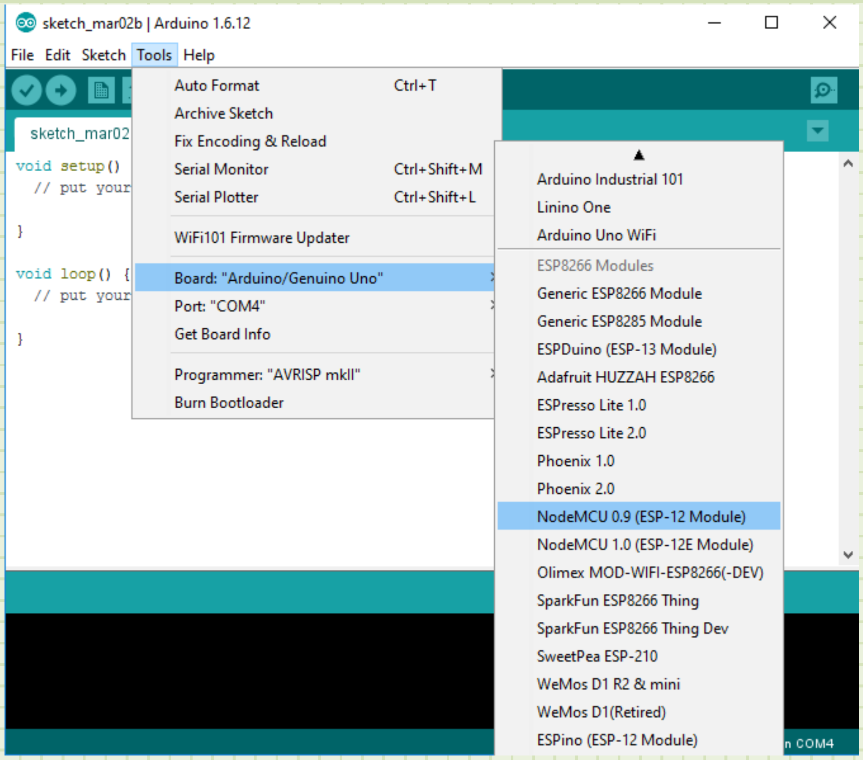
SparkFun AVR Boards by **SparkFun Electronics** version **1.1.5** INSTALLED
Boards included in this package:
RedBoard, MaKey MaKey, Pro Micro, Fio v3, Qduino Mini, Digital Sandbox, Mega Pro, RedBot, Serial 7-segment Display, ATmega128RFA1 Dev Board.

Click on the "Install" button. When the install is complete, click the close button.

We are really close to talking to FRED. We just need to select him as the active board type.

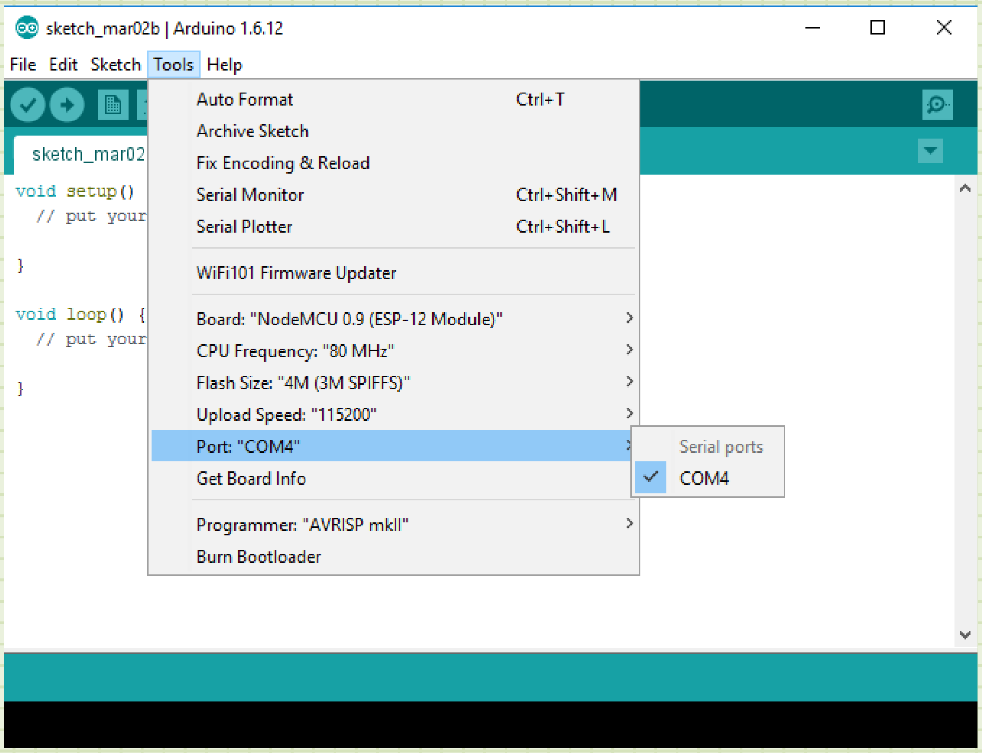
Select **Tools->Board:->NodeMCU 0.9 (ESP-12 Module)**

You may need to scroll down the list in order to find this. ESP-12 is a variation of the 8266 chip. This is the communication computer chip that FRED uses. It is a very popular chip for IoT applications because it includes a computer, Wi-Fi radio, and input/output controls on a single chip. A lot of times, chips like this are referred to as System on Chip (SoC). These are single small integrated circuits that have a lot of circuitry to support the development of devices. Often complex devices can be made from this single chip.



After you select this board, the software then knows what to communicate with.

Take out FRED. Plug in the USB cord to the lower USB connection on FRED. Plug the other end into your computer. Your computer may take a second to load the drivers for this new USB device. When it is done, you should be able to see the port that FRED is connected to.

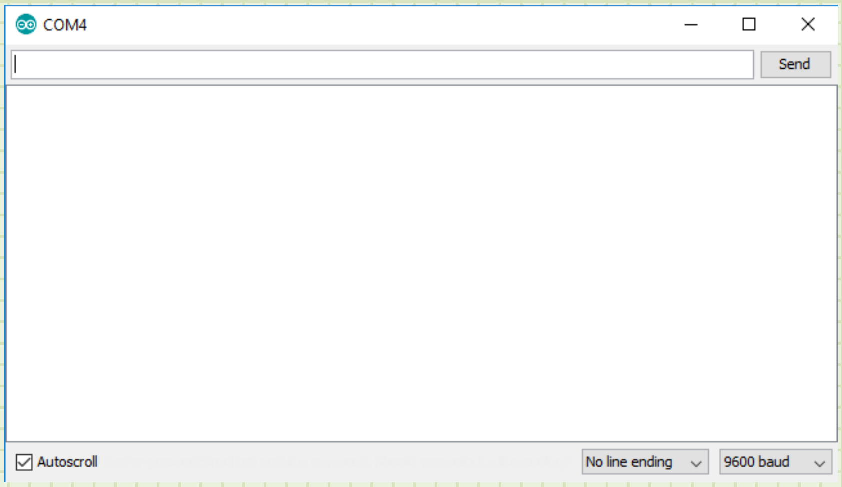

sketch_mar02b | Arduino 1.6.12
File Edit Sketch Tools Help
sketch_mar02
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "NodeMCU 0.9 (ESP-12 Module)"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Upload Speed: "115200"
Port: "COM4"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM4

```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Select **Tools->Port**

The “Port:” field should have the word COM with a number following it. This is a serial communication port. It is a port the computer reserves for this device. In the above example, FRED is connected as port COM4. If there are multiple options, you may need to mouse over the **Port:** menu item and select the COM port.

When you have the port selected, on the same menu click the selection that says “Serial Monitor.” This is a console that you can communicate over to talk to FRED. You should see the following:



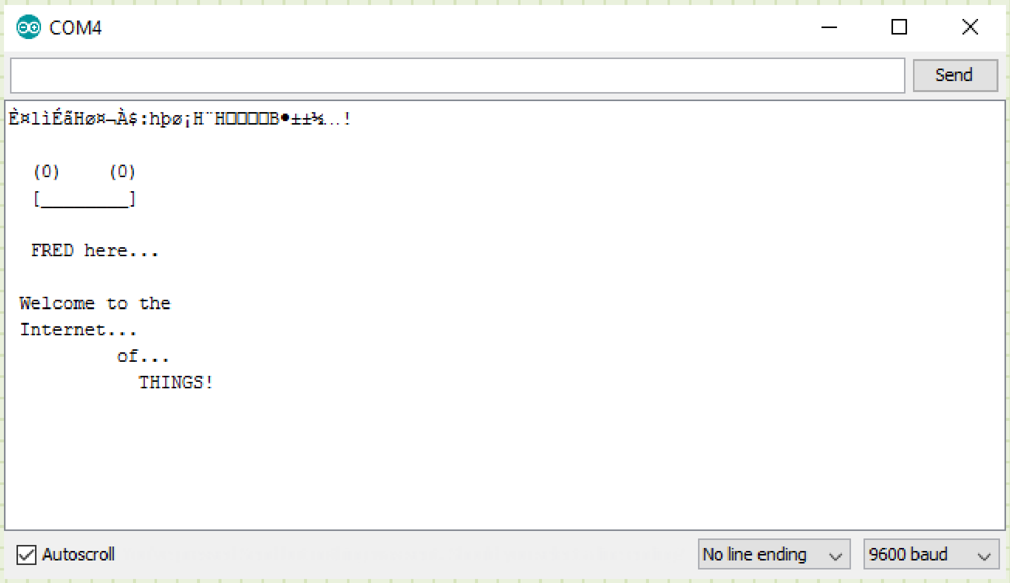
Autoscroll ✓

No line ending

9600 baud

You may see some random information on the screen. Ensure that "9600 baud" is displayed in the box on the lower right. If it is not, use the dropdown arrow to select it.

Press the RST button on FRED. You should see the following:



```
È¤liÉãHø¤¬À¢:hþø¡H¨H□□□□□B●±±¾..!


  (0)      (0)
  [_________]


  FRED here...

 Welcome to the
 Internet...
          of...
            THINGS!
```

Autoscroll ✓

No line ending

9600 baud

Yea!! You are talking to your new buddy. We will be able to program him now. FRED comes loaded with this initial message built in. When we program him from here, you will write over this code. Even though you won't see FRED anymore, know he is working hard for you.

Blynk Setup

Up to this point, we have setup the Arduino software and made it able to communicate with FRED. Now we will setup the Blynk libraries and mobile app to finalize our IoT connections.

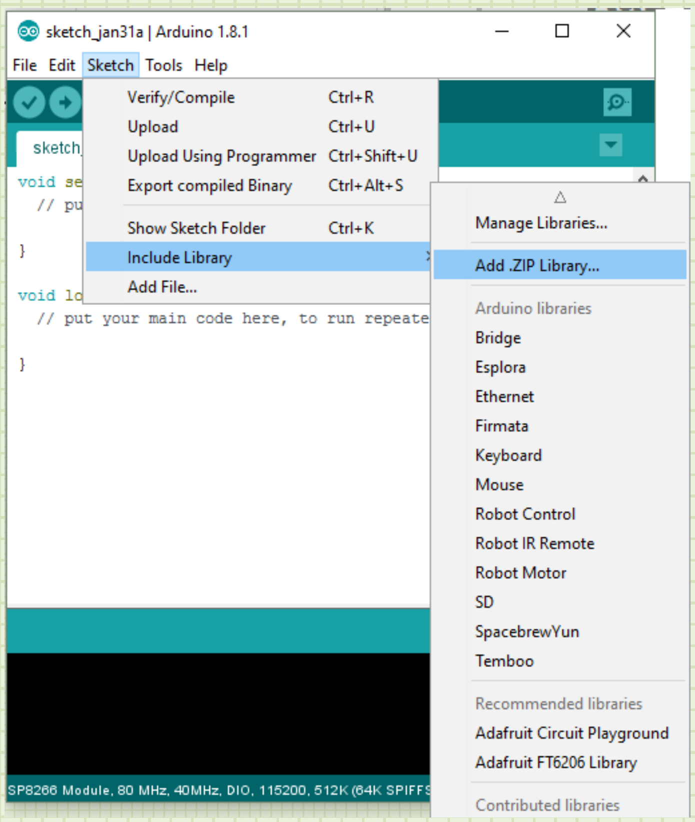
Library Download and Installation

The first step is loading the Blynk libraries into the Arduino software. This is a much simpler task since Arduino changed the way it imports libraries. We only need to download the library from the internet in a .zip format and then have the Arduino software load this zip file.

Go to the Subsystem download page (www.subsystems.us/files.html). Download the file blynk.zip from the Subsystem 3 section by clicking the file. Make note of where you download the file to.

With that file in our possession, we can now load it into the Arduino software. Open the Arduino program on your computer.

Select **Sketch->Include Library->Add .ZIP Library**



```
void se
  // pu

}

void lo
  // put your main code here, to run repeate

}
```

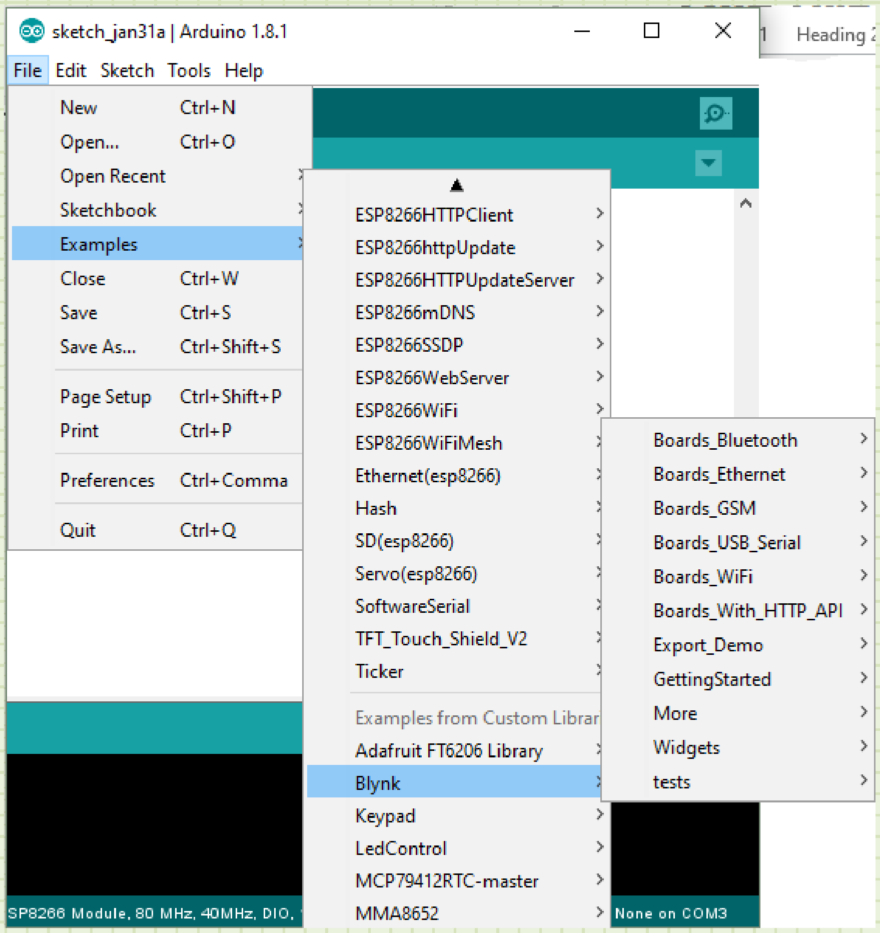
```
SP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS
```

You will then be presented with a file explorer. Use it to find the .ZIP file you just downloaded. When you find the file, double click it, or select it and press the "Open" button at the bottom of the dialog.

Arduino will take this file and unzip it, copy the contents to the correct folders, and load sample programs.

If all went well, the library loaded and you now have some Blynk sample programs. Let's check that to make sure.

Select **File->Examples->Blynk (**you may need to scroll down to find it)



You should see a bunch of sample applications in this folder.

Excellent. Now let's get the app on your smart device.

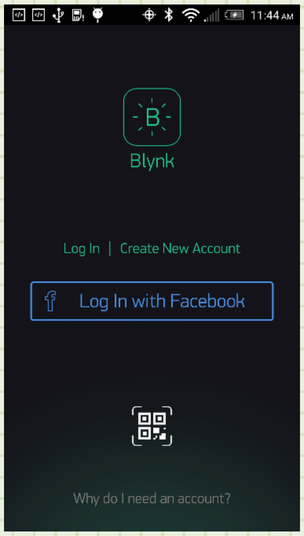
Loading the Blynk App

The Blynk app will load on either Android or Apple devices (smartphones and tablets). Choose which device you want to use. You can load the Blynk app using either of these methods:

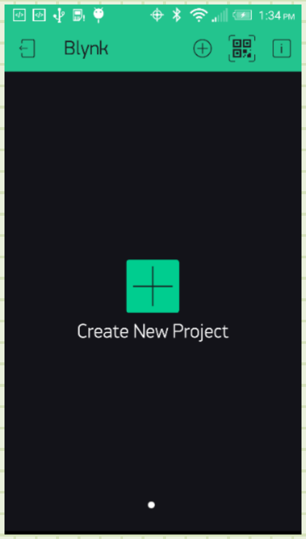
1. There is a good chance that you already know how to download apps from the Google App Store or from iTunes. If you are comfortable with this, go to your applicable app store, search for "Blynk", and install the app on your device.

2. If you are not sure which app to download or are having problems finding the app store, go to the Blynk Home page (www.blynk.cc) on your device and select the "**Getting Started**" menu item as you did before. When you get to that page, there will be links for the app based on whether you have an iPhone or an Android powered device. Click the appropriate button and you will be taken to the correct store and correct app. Download the app to your device.

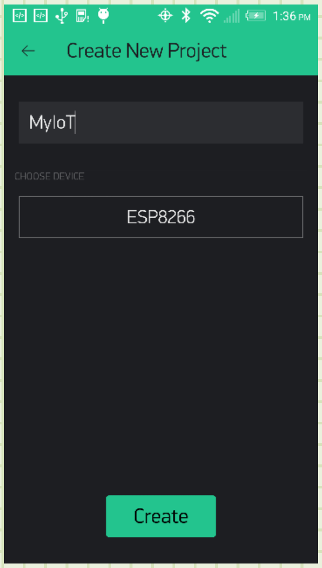
Once the app is downloaded, open it.



You will be greeted with a sign-on page. If you have never used Blynk before, you will need to create an account. Blynk is a business, but they allow people to use a limited amount of their services for free. Having an account will also give you access to all of your projects on any device you want to use. There is also a long code that your app will generate that you will need. By putting in the correct email address, Blynk can send you this access code instead of having to write this down. This also gives you the ability to log in with your Facebook credentials if you prefer. After you log on, you will be presented with the main screen. You may also be presented with a popup screen introducing new features or helpful hints. You can just close that and continue.



This is a summary screen of all of your projects. You don't have any right now, so there is not a lot to look at. When you have multiple projects, you will able to swipe through them. Let's get working on those projects! What we need right now is an access code. When you create a Blynk project, you will automatically be emailed the access code for it. This is the security code used to ensure that your device only responds to requests from authorized users. It is a long alphanumeric code so it is a nice feature that it is emailed to you. This way you can cut and paste it into code. Click on the "Create New Project" button to start a project and get the access code. This code is referred to as an **authorization token**.



This brings you to your project creation page. Here you can give your project a name and assign a particular class of device to it. Type in the name "**MyIoT**" and select device **ESP8266** if it is not already selected by pressing the box and using the pop up dial selector. Press the "**Create**" button at the bottom of the page and you have done it. You created a project in Blynk. You need your authorization token. Check your email to ensure you received it. If you haven't, you can get it by selecting Project Settings (the icon that looks like a nut in the top menu of your project) and tap on the device at the bottom. This will bring up the My Devices screen where you will have the ability to resend the authorization token by email or copy it to the clipboard by tapping it. Then you could paste it in an email or as a last resort, paste it into a note and then copy the number down. Hopefully, you received the email from the Blynk service and don't have to manually do this. Make sure you have this authorization token because we need it to finalize the information we need to program into FRED. Let's do that now.

Section

5

Program FRED

You have come a long way. Let's review really quick where we are.

1. We loaded the Arduino software so we could program FRED
2. We connected FRED and received a message from him letting us know he is ready for our code
3. We loaded the Arduino software with the Blynk libraries needed to communicate with the Blynk server
4. We loaded the Blynk app on your device and started a project
5. We received our authorization code

That was an awful lot of setup. But we are now in a position to finally program FRED with the base code, Blynk code, authorization code, and your Wi-Fi information. He will have everything he needs to hook into your network and communicate with the Blynk server. From there, he can then respond to our requests with the Blynk app. Here are the final steps:

1. Download the code to FRED
2. Add widgets to the Blynk app to access FREDs input and output

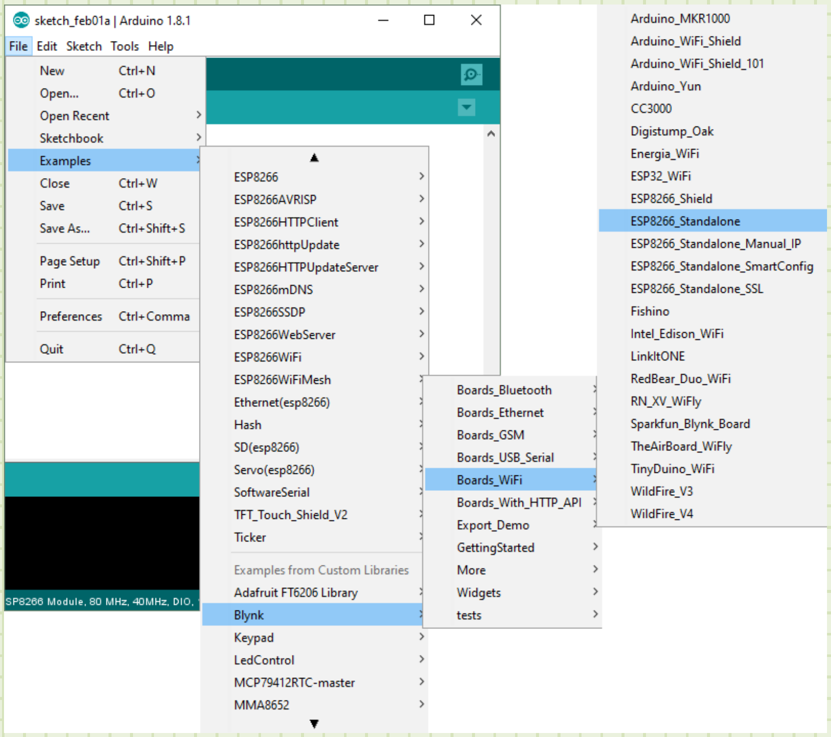
You will need the following information to do this:

1. Your authorization token
2. Your Wi-Fi connection name (you can usually check this on your phone by looking at your Wi-Fi section and seeing what you are connected to)
3. The password to your Wi-Fi network

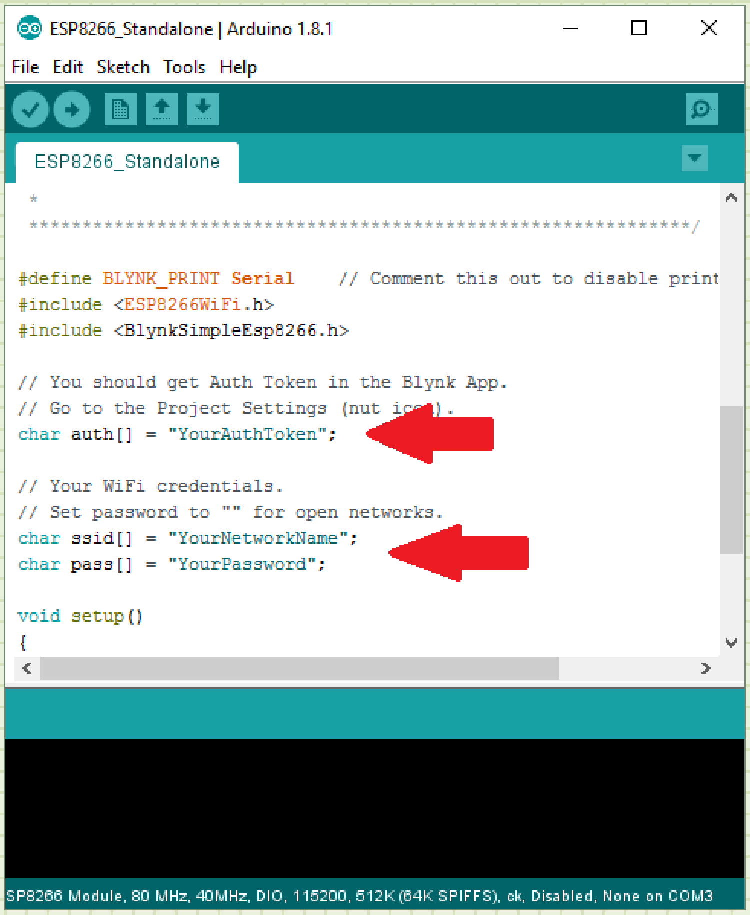
With this in hand, we are ready to talk to FRED.

Open the Arduino software application. From here, we are going to take advantage of the built in example programs that were installed when we installed the Blynk library.

Select **File->Examples->Blynk->Boards_WiFi->ESP8266_Standalone**



This example application has all of the structure we need to communicate with our board. When the program loads, scroll down a little until you see the places marked by the red arrows in the figure below. This is where we will place our specific information.


ESP8266_Standalone | Arduino 1.8.1
File Edit Sketch Tools Help
ESP8266_Standalone

```
 *
 **************************************************************/

#define BLYNK_PRINT Serial    // Comment this out to disable print
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

void setup()
{
```

SP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS), ck, Disabled, None on COM3

On the line that says char auth[] = “YourAuthToken” replace the text “YourAuthToken” with the authorization token sent to you by Blynk. The easiest way to do this without making a transcription error is to cut and paste it from the email you received. Make sure you enter the authorization token with quotation marks at the beginning and end.

On the line that says char ssid[] = “YourNetworkName”, replace the text “YourNetworkName” with the name of your own Wi-Fi network. Again, make sure you enclose the name in quotes. Note that this is case-sensitive so make sure you enter this correctly.

Finally, on the line that says char pass[] = “YourPassword”, replace the text “YourPassword” with the password of your own Wi-Fi network. Again, make sure you enclose the password in quotes.

Now that your information is entered, save this file so you can always recall it and reload it without having to reenter this information.

Select **File->Save As…**

The dialog that pops up is the typical save dialog. Select a place to save the file and a name. You may want to call this “MyIoT” or something similar so you recognize it. Click “Save” when you are done.

Let’s compile this application to ensure the information was entered correctly.

Select **Sketch->Verify/Compile**

You will see a progress bar and the message “Compiling sketch…” in the message area. If all goes well, you should see the program compile and receive a message in the message area that says “Done compiling.” If there are any errors, ensure you entered the information above correctly, fix any problems, and try to compile again.

Getting FRED Ready

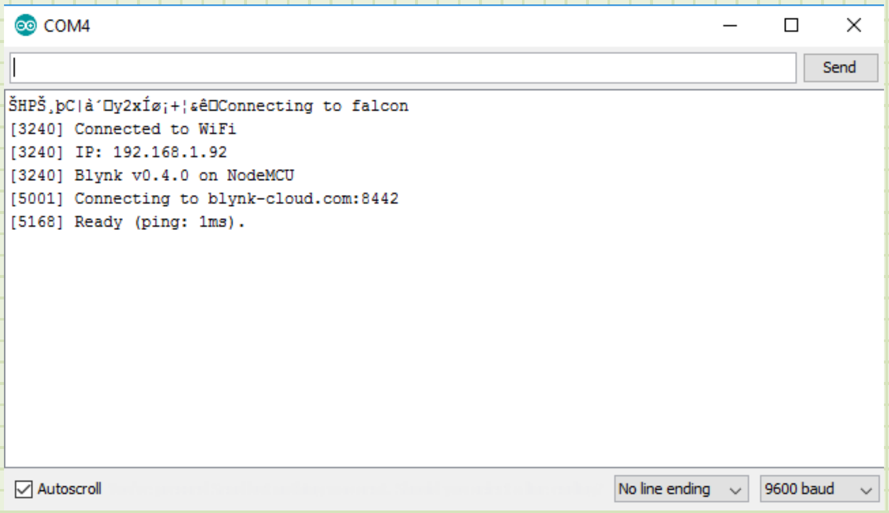
With the program compiled and ready, we can now get FRED ready to receive these instructions.

1. Plug FRED into your computer by connecting the USB cable to your computer and the lower USB port on FRED.
2. In the Arduino program check to make sure the port FRED is connected to is selected (Select Tools from the menu bar and see that Port: has the correct entry)
3. On FRED, simultaneously press the RST and FLASH buttons. Then release the RST button, wait 2 seconds, and release the FLASH button. This puts FRED in programming mode.
4. In the Arduino software, select **Sketch->Upload**. This will recompile the program and start transferring the program to FRED.
5. Open the Serial Monitor (select Tools->Serial Monitor). When the upload is complete, you will see FRED report his connection to the internet and to the Blynk server.

The upload will take a little while. You will see a progress bar on the Arduino program and a little blue light on FRED will flicker. The Arduino software will report when the download is complete.

We verified that communications were setup correctly before when you hooked up FRED and saw the welcome message so this process should go smoothly. If there are issues, make sure you have the correct board selected and the correct port. Also, try to place FRED in programming mode and try to download again.

The serial monitor will display the connection details. You should get text in the serial monitor that looks like the text in the figure below. YourNetwork should show the network name that you entered into the program. If you missed this text dump, make sure that the Serial Monitor is running and press the RST button on FRED. This will reinitialize that process and you should see the text now. If you don't see an IP address (IP:), then FRED had trouble connecting to the Wi-Fi. Check your Wi-Fi settings in the program. If you don't see the Blynk message, check your authorization token. After you correct any problems, go through the download steps again.



```
ŠHPŠ¸þC|à´□y2xÍø¡+¦&ê□Connecting to falcon
[3240] Connected to WiFi
[3240] IP: 192.168.1.92
[3240] Blynk v0.4.0 on NodeMCU
[5001] Connecting to blynk-cloud.com:8442
[5168] Ready (ping: 1ms).
```

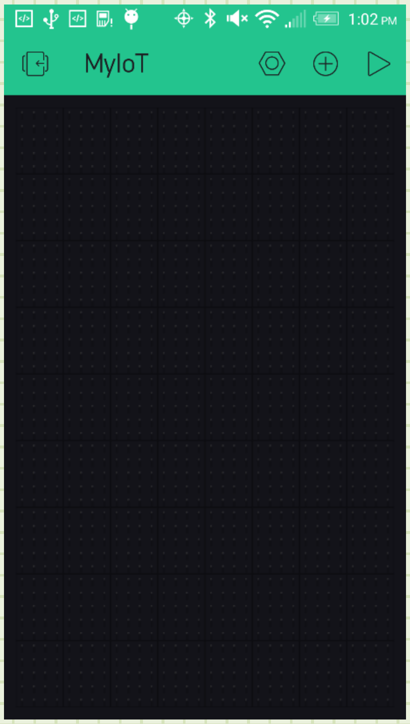
That was a lot of effort, but if all went well you are done with FRED's configuration. He is now a standalone computer hooked to the internet and the Blynk server. You can now disconnect him from your computer and just use the USB port to power him from a computer USB, USB charger, or phone charger with a micro USB connector. As long as he is in your Wi-Fi range (the Wi-Fi you set him up to), he will connect up and start talking to the Blynk server.

Section

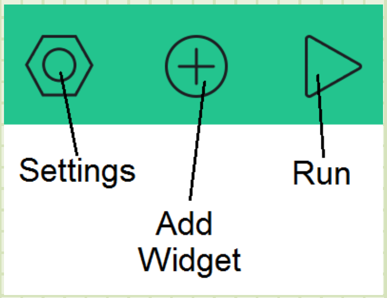
5

Customize the Blynk App

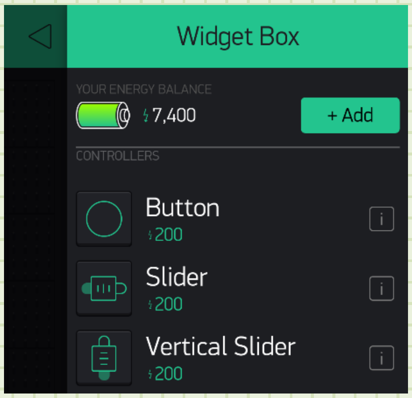
We have worked hard to get FRED hooked up to the internet and talking to the Blynk server. Now, for the really fun part. Customize our interface with the Blynk app to send and receive information from FRED.



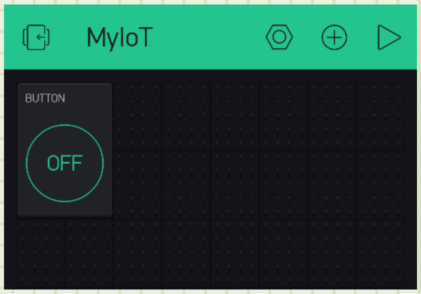
Open the Blynk app on your device. If you are in the project browse area, swipe to get to your project and tap it to enter edit mode. The app will start you on a blank canvas. You can add what are called "widgets" to the canvas to interact with devices that are hooked up to the Blynk server. As luck would have it, you have a device hooked up to the Blynk server so let's add some widgets and interact with it. We are going to start by adding a button to cause the Red section of the LED light up.



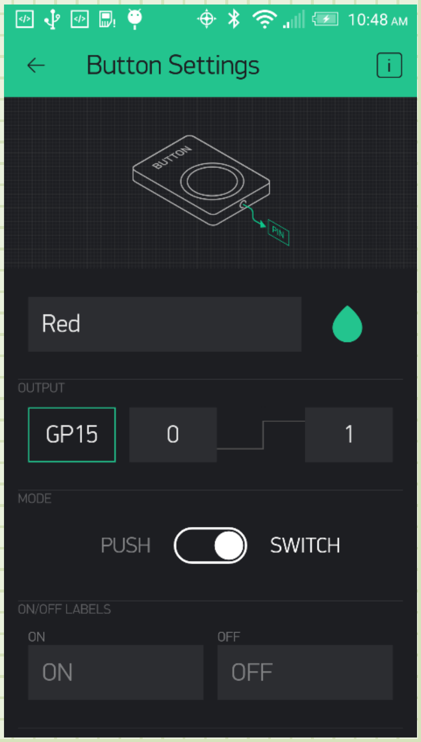
On the top of the project screen there are some icons at the top. Settings allows you to modify your project settings and view your authorization token. Add Widget open a page to add functionality to your project. Run changes from edit mode to connected and running.



Click on the Add Widget menu item to open the widget menu. The first widget we will add is a Button. A Button is a way to send and off/on signal. We will use this to light the Red LED. Tap the Button widget. This will add it to your project.



The Button appears with some generic information. We want to customize this so it will reflect our system. In the Edit mode of our project, if you tap a widget, the options for that widget will be displayed. Tap the newly created Button.



Red

OUTPUT

GP15 0 1

MODE

PUSH SWITCH

ON/OFF LABELS

ON: ON

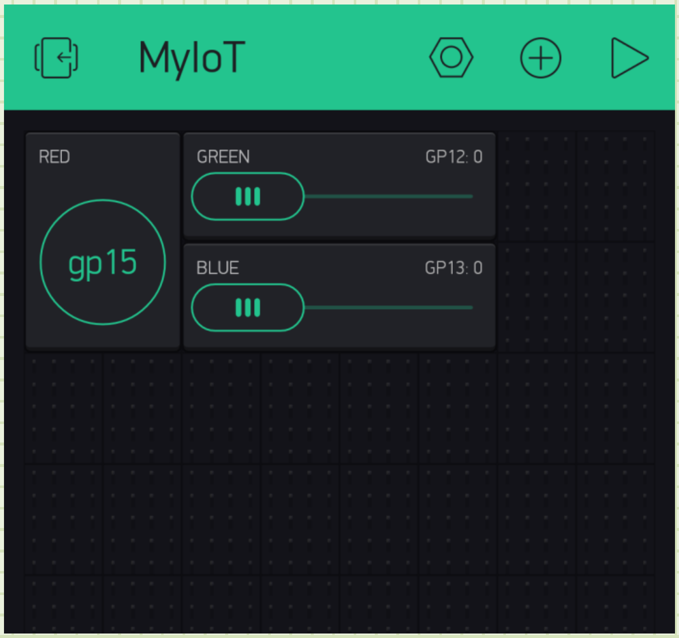
OFF: OFF

On this page, we will add a name for our Button and tell the app what port of FRED we want this button to operate. The Red LED on FRED is connected to General Purpose pin 15. In the Name box, type in the word **Red**. This will label the button Red so we will know what it controls. In the Output section, click on the PIN box to bring up the pin select dialog. Use the controls to select **Digital** pin **gp15**. Select the Mode to be **Switch** and click Continue. Your screen should look like the one to the left. Tap the arrow next to the page title (Button Settings) to get back to your project page. You should now see that your button is labeled Red and the button should show the gp15 pin we selected.

The time has come. Tap on the Run menu icon to run this project. Are you ready? The fruits of all your effort is at hand. Make sure FRED is plugged into a power source and has been given a minute to link up with your Wi-Fi and the Blynk server. Now tap your Red button.

If all is well, FRED will respond by lighting the Red section of the LED. Tap the button again to turn it off. Now, wherever you are in the world, tapping this button will light this light. **You are now a part of the Internet of Things**.

Let's do some more. Let's add two more widgets to control the other two colored LEDs. Tap the upper right corner icon to exit Run mode and go back into Edit mode. Click the Add Widget icon to get to the Widget page. This time, tap the Slider widget to add a slider. Do this again to add a second slider to your project. Tap on the first slider. In the Slider Setting page, enter **Green** for the label and select **gp12** for the **Digital** Pin. Go back to your project page, select the second slider and set the label to **Blue** and the pin to **Digital gp13**. When you are done, your project screen should look like the following:

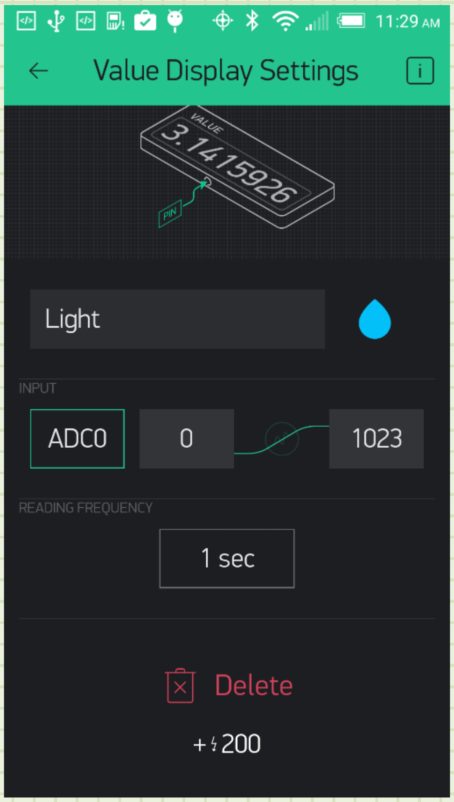


Sliders operate by sending a varying value to the connected device. FRED will use the value set by the slider to control the brightness of the Green and Blue LEDs. This will allow you to create different colors. Run this project and move the sliders. See the response by FRED.

You have programmed a typical IoT application. This application is essentially controlling lights in your house. If you hooked up the outputs of FRED to some control circuitry, the same application could be controlling the actual lights in your home. The combination of turning the Red LED on and off and using the sliders to control the Green and Blue LEDs will allow you to make all sorts of different colors. This could be used to let those at home know what mood you are in or even allow you to set different colors to represent different states. You might turn on the Red when you are at school or work. Turn on Blue when you are on your way home. You can come up with all sorts of uses for this simple device.

Let's look at another application.

Instead of sending information to FRED, what if we wanted to get information from him? FRED has a built in light sensor that samples the ambient light and reports it to the Blynk server. Let's add a widget to display these readings.



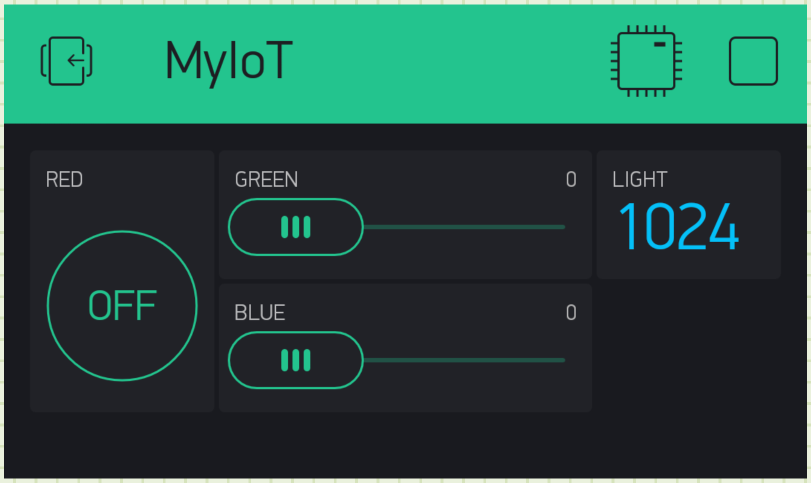
Light

ADC0 0 1023

1 sec

Exit Run mode and enter Edit mode. Now, tap on the Add Widget menu item. This time, add a **Value Display** widget. You may need to scroll down the list to find it. Once you have added that to your project, tap on it to open the settings screen. Give this widget the label "**Light**" and select **Analog adc0** for the pin using the selection wheels in the pin select section. You should have a settings screen that looks like the one at left.

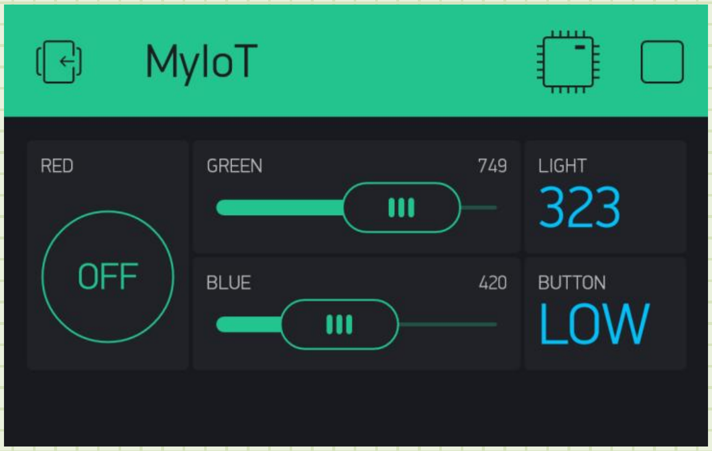
When you run this, you will still have control of the LED as before, but now you have an added function of reading the light level detected by FRED. This will be a number in the Value Display widget. This number will go from 0 to 1024 based on the amount of light. Put your hand over the sensor on FRED and watch the display change.



This is another great application for the IoT. With this, you may be able to tell if anyone is home at your house at night. If the sensor shows a high value, there is a lot of light in the area. If it is low, there is little light. This type of remote monitoring of sensors is a common application for the IoT. Farmers can monitor the moisture in their soil, track livestock, check river water levels, etc. There are many industries that benefit from this ability to quickly monitor remote parameters. You have created a pretty sophisticated connected sensor platform.

Let's add one last function. There are times that you want to have an input on your device that sends data to the cloud. FRED has a button hooked up to General Purpose pin 4. We will use this to show on our Blynk app when someone presses that button.

Take your project out of Run mode and place it back in Edit mode. Tap the Add Widget menu item and add another **Value Display** widget to your project. Tap on the widget to bring up the settings screen. Give it the label "**Button**" and assign it to **Digital** pin **gp4**. Return to the project page and run your project. The digital pin that FRED's button is attached to shows a HIGH voltage when it is open and a LOW voltage when pressed.



The display should be showing HIGH right now. Press the user button on FRED (single button on the top board) and hold it. You will see the Value Display show LOW. There may feel like there is a delay for this button to create the HIGH and LOW indications and you would be right. FRED is reporting the status of this button every second to the Blynk server. If you click and release it in between these report times, your app won't sense it. This type of reporting is often referred to as **broadcasting**. FRED isn't waiting for us to ask for information. He is just broadcasting this information to the Blynk server every second. The other method that can be used is called **polling**. With polling, FRED would just sit there until he received a request for data and then would transmit it. These are two very different but important data exchange concepts used throughout the communication industry. Since FRED is set up to broadcast the data, make sure you hold the button for at least a second to ensure the button press is seen within the broadcast period.

This button is very similar to the button companies now offer to order products from the point of use. When you run out of your favorite detergent, you can now just press a button on your washer to order more. This could also be a remote button that a person carries so they can press it if they need emergency help.

FRED is now broadcasting his information over your Wi-Fi and the internet to the Blynk server. Your phone is interacting with this data by receiving information over the internet and then Wi-Fi or Cell phone data links.

We encourage you to look at the list of resources in the Appendix to continue learning about Arduino, Blynk, and other IoT devices.

Welcome to the Internet of Things.

Appendix

For more information on Arduino, visit www.aduino.cc

For more information on Blynk, visit www.blynk.cc

For more Arduino and IoT kits, visit www.sparkfun.com

www.adafruit.com

List of IoT devices for purchase, visit www.iotlist.co