Comparing interface and application programming tools

FABLAB BRIGHTON 2018

This week we're comparing a few tools, including:

- WeMos D1 mini (Pro) and Node-RED
- Grasshopper and Firefly
- Python

WeMos D1 Mini (Pro) and Node-RED



© DanAsker

The <u>Wemos mini</u> is a miniature wireless microcontroller development board that can be programmed via the Arduino IDE. Like the Arduino boards i has input/output pins, but gains extra functionality with the wifi chip and antenna.

<u>Node-RED</u> is a visual programming tool, not dissimilar to how grasshopper works, with 'nodes' of code that can be put together and connected via 'wires'. Each node has a purpose that takes data, does something with the data and then passes it onto the next node.

Firefox File Edit View History Bookmarks Tools Window Help	÷ 📬	↓ • >	🖇 🤶 📢 52	% 💼 🛛 Wed 20	:49 Q :
🔍 🔍 🔍 Video: 🔯 Mail- 🖌 High-i 🥸 Arduir 🤷 How t 😯 Sign t 🦉 Mail- 😯 Sign i 🚱 Sign t 🌋 Applic 🕿 No X	А Р	robla 😤 Nac	ie- 💭 NodeF	> + ~	
← → C' û ① ▲ https://testing3.eu-gb.mybluemix.net/red/#flow/4a24b170.de5c08	··· 🛛		\mathbf{F}	II	
🌣 Most Visited 🝍 Raspberry Pi • View 🜐 3D Scanner 🔟 Fab Academy Assig 🜐 Fab Classes 🚱 My FabAcademy Site					en Shi
Node-RED		-	🗧 Deploy 👻	*	=
Q filter nodes Flow 2	+	info	debug	dashboar	x
v input		~ Informa	ation		en Shi
⇒ iniect ■		Selection	9 nodes		620.:
					-
					en Sh
					60.
link					
() mqtt					en St
http 1 10 LED Colour - f Get Red, Green, Blue values - f Format Data Open th	is and s	-			60.
Websocket					
top top					en Si
mqlight make Stepper Steps Format Data					60.
🔁 bmiot 🗘					Lever
x autout					on Si
		You can	confirm your	changes in th	60.
debug		node	edit tray with	Henter or	in pick
		cano	ol them with		and the second se
link			sei utern wiur	<i>mescape</i>	
matt ()			Ser them with	#escape	en Sh 620.

Node-RED lends itself to interface application and can be a much more accessible way to gain this skill than learning a new coding language. The platform is open source so users can create their own nodes using code if they want.

We had a lesson from Tim Minter of IBM. We started with a kit that had a Stepper motor, a servo and 3 RGB LED NeoPixel strip all hooked up to the Wemos Mini providing power and digital input/outputs.

This is a description of what we did. It shows how complicated it and what's involved. IBM is in the process of changing from Bluemix to IBM Cloud and the steps are also being updated. Hopefully this will make it more intuitive to use. At the time of writing these details are more up to date than many of the tutorials out there, but maybe not for long. We are still at the beginners stage of a steep learning curve, but once beyond that it will make connecting devices to each other and the web easier.



We dived straight into the Node-RED environment. We used the cloud option for this and created a free IBM Creative Cloud account at www.ibm.com/cloud. After confirming the account via email we selected the Catalog menu in the top right hand corner and when the page opened we scrolled down to Boilerplates and installed the IoT platform starter.

Note: It would have seemed logical to install the Node Red starter but don't because it doesn't have the devices panel for attaching things.

🌣 Most Visited 👋 Raspberry Pi •	View	Classes 🛛 🖓 My FabAcademy Site		4 <u> </u>	0 0 -
≡ Ö IBM Cloud			Catalog	Docs Support M	lanage 🧿
	Cloud Foundry apps /				
Getting started	testing3 🔶 Running	Visit App URL		Routes	•
Overview	Org: paters936 Location: United Kingo	dom Space: play ground			
Runtime					
Connections	Runtime				
Logs	\frown	\frown	\frown	\bigcirc	
API Management	#		⋳ 256 ⊕	(256)	
Monitoring		ų – j			
	BUILDPACK	INSTANCES	MB MEMORY PER INSTANCE	TOTAL MB ALLOCATION	
	Internet of Things Platform Starter	All instances are running Health is 100%		510 GB still available	
	Connections (2)				
	testing3-cloudantNoSQLDB				
1	testing3-iotf-service				



We went onto the page and gave our application a name and clicked on Create at the bottom right. It then starts creating a database and installs the instances that we need (ie it is giving us 3 servers - IoT, Cloudant database, and Node Red). This takes a while. When it's done the word 'starting' at the top changes to 'running'.



Next to the word 'running' we clicked on 'Visit App URL'

And a new tab opens with a welcome page. We clicked on 'next', created a username and password and clicked 'next' and 'next' again

-) → C ^a ¥ Most Visiter			
Most Visited	<u>۵</u>	🕐 🔒 https://testing3.eu-gb.mybluemix.net 🚥 💟 🏠	⊻ ⊪∖ ۞ ◘ =
	Respberry Pi • View	O Scanner III Fab Academy Assig D Fab Classes IV My FabAcademy Site	e G
		Welcome to your Internet of Things Platform (IoTP) hollernlate	
		application on IBM Bluemix	10
			6
		This sample application uses Node RED to help demonstrate the wonderful things you can do with your InTP service. We know you're eager to check it out, but first there is something important to do:	1
		Secure your Node-RED editor	6
			-
			10 8.
			ie 5
			ie.
			3.
		Previous Next	6.
refox File	e Edit View Histo K 119 Video 🔀 Fa	Roferantis str.p	 ■ 58% ■> Wed 20:33 Proble > + ~
refox File ● ● → C ⁴ Most Visited	Edit View Histe	Artificial La Construction de la	 ≤ 55% ■> Wed 20:33 A Proble > + ~ ★ IIA ② □ =
refox File ● ● → C ⁴ Most Visited	Edit View Histe Ras Video Ras Raspberry Pi • View	Artificial Sector Action File Field Action File	 saw ■> Wed 20:33 Proble > + ★ IIN ③ □ =
refox File ● ● → C ⁴ Most Visited	Edit View Hists Ing Video	Order strip, strip, strip, Dockmarks Tools Window Help Image: Strip (Strip (Str	 S5% ■) Wed 20:33 mod 2) + + ± In (2) (0) =
refox File ● ● → C ⁴ Most Visited	Edit View Hist A 18 Video 🔀 R C Raspberry PI • View	order Image: Secure your Node-RED editor	 € 58% ■) Wed 20:33 A Precis > + ★ + ★ III <> C
refox File	Edit View Hists As View Eight As View Eight As View Eight	Artification of the second sec	 ■ 58% ■D Wed 20:33 ↑ Proble > + ★ + ★ IN © CD =
● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	Edit View Hists fig View Mister fig View fig View fig View fig View	offer str.p. Image: Source	 sex ■> Wed 20:33 reck > + ± + ± IIN O CO =
Interfox File	Edit View Histe (ng view	original served by Window Help Image: Contract Served by Window Image: Contract Served by Window Help He	
irefox File ● ● → C ⁴ Most Visited	Edit View Histor R8 Video & R C C Raspberry Pi • View	organ Sockmarks Tools Window Help Sockmarks Sockmarks Tools Sockmarks Sockmarks Tools Sockmarks	 sex ■) Wed 20:33 meei > + ± ■ ± IIN > 00 =
refox File ● ● → C ⁴ Most Visited	Edit View Hists	organ Organ <td< td=""><td> €5% ■> Wed 20:33 Moci > + Wed 20:33 Moci > + Wed 20:33 The set of the set</td></td<>	 €5% ■> Wed 20:33 Moci > + Wed 20:33 Moci > + Wed 20:33 The set of the set
Herefox File	C Edit View Hists C 16 Vize.	order Image: Solution of the sol	 58% ■) Wed 20:33 Mecil > + ★ Max 10 ★ Max 20 Max 20<!--</td-->
refox File	Edit View Hists C D View R r C D View R r R rapberry PI - View _	order Image: Control of the control	 € 58% ■) Wed 20:33 A receil > + ★ 10 ★ 10 ★ 10 ★ 10 ★ 10
refox File ● ● → → C ⁴ Most Visited	Edit View Histor November 1 Review 1 Review 1 Review 1 Review 1 Review 1	org Ookmarks Tools Window Help Ookmarks	● 55% ID Wed 20:33 A Proce > + ↓ ↓ IN ② CD Ξ
refox File → → C ⁴ Most Visited	Edit View Histor R Video K 7 C Table State Raspberry Pi + View	org Occumarka Tools Window Help Occumarka Occumark	 55% ■) Wed 20:33 Arrow > + ★ IN ○ ID Ξ
refox File → C Most Vialted	Edit View Histor R Video	orgetting Image: Contraction of the contrac	 sex → Wed 20:33 A rec: > + → ↓ HN > D =
refox File	Edit View Histor Rit Videe & rit Rit Videe & rit Ritspherry Pi+Vew	orgenerative Tools Window Help Image: Control of the control of t	 sev → Wed 20:33 tree: > + → ⊥ III > 0 □ =
Firefox File	Edit View Hists C 10 View	ary Solution Network Solution Network Solution Network Solution Network Solution Network	 58% ■> Wed 20:33 Mecei > + In In In
Firefox File	Edit View Hist C N2 View X R C X R Raspberry Hi - View _	any Secure your rolder so only authorised users can access it Username Password Or Allow anyone to view the editor, but not make any changes Not recommended: Allow anyone to access the editor and make changes	● 55% ID Wed 20:33 A Proce > + ↓ ↓ IN ○ ID =
frefox Filder	Edit View Histor November 1	org Ook Window Help Ook	 55% ■) Wed 20:33 Proz. > + ∨ ⊥ IIA ○ □ Ξ
refox File	 Edit View Hists RØ Video RØ Video RØ R Raspberry Pi+ View 	org Occumarka Tools Window Help Occumarka Occumark	 ssx ■) Wed 20:33 mea: > + w IN > D =
refox File	 Edit View Hists RØ Video RØ Sideo RØ Sideo Raspherry Pi + View 	org O	 sex → Wed 20:33 Arrea > + → ⊥ IN > D =

This installs our user credentials and finishes the installation of Node Red

Then we clicked on the button to take us to Node Red and put in the user name and password.

It loads with a sample Flow.



The Node-RED environment consists of inputs/output and data processing nodes on the left, a workspace in the middle and a menu on the right where you can debug and view what is happening to the data at the the different stages of the sketch. LED

We introduced a 'colour picker' node from the dashboard tab for our LED. This connected to a 'function' node from the function tab which allowed to to enter code to extract the RGB data.

// msg.payload is what "comes into" this node. This is the same for most NodeRED nodes
var colourInput = msg.payload;
// In this case msg.payload comes in looking like this "rgb(100,100,100)" where the
// In this case msg.payload comes in looking like this "rgb(100,100,100)" where the

// values in the brackets ate the amount or red, green and blue light respectively

// (on a scale of 0 to 255).

// That input format is no good for us, we need to extract the individual red,
// green, blue values...

//first lets remove the beginning "rgb(" part of the text colour = colourInput.replace('rgb(', ''); //this replaces 'rgb(' with '' ie nothing // now lets replace the end bracket with nothing... colour = colour.replace(')', ''); // finally we remove all the spaces colour = colour.replace(' ', ''); // now the colour vaiable contains just the red, grren, blue values separated by

// commas eg "100,100,100"

// To do anything with these values we need to put them in their own variables

// We do this by creating an array of the values which is straight forward now that

// we have them in a comma separated string

var colourArray = colour.split(","); // we create an array and use the split function

// to put each value into the array.

// now we can get to each value using the colourArray[x] notation where x is the

// position in the array we want to look at. The vluae have ben put into the 0,1,2 // positions in the array

msg.red = colourArray[0]; msg.green = colourArray[1]; msg.blue = colourArray[2];

// now when a rgb value comes into this node, the node outputs the individual // red, green, blue values which can be accessed by looking at // msg.payload.red, msg.payload.green and msg.payload.blue values

return msg;

After the function node we went into a 'template' node, also in the function tab which contained;

"R":{{red}},

```
"G":{{green}},
"B":{{blue}}
```

All the data was output as 'Parsed JSON'. The final connection was to the IBM IoT node found in the output tab. To save and activate our nodes

Node red sounds simple but It doesn't just connect to the device.

We had to create a device by clicking on 'Devices' on the left, then 'Add Device' on the top right. Then we added a device type and device ID in the boxes. We called it Wemos for both.



BM Watson IoT Platform	QUICKSTART SERVICE STATUS DOCUMENTATION BLOG ion®youonly.co.uk ▼ ID: (8c7dxa)							
ਗ਼ ∢ Back								
DEVICE DRILLDOWN	You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.							
A Connection Information Recent Events	Organization ID 8c?dxa							
Device Information	Levoz lybe wernos Device ID wernos Authentication Mehod use-token-auth							
Metadata Extension Configuration	Authentication Token wernoswernos							
Diagnostics Connection Logs Device Antions	Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.							
	Find out how to add these credentials to your device A							
	Connection Information							

On the next 2 pages we just clicked 'next'.

For the authentication token we called it wemoswemos because it had to be at least 8 characters and clicked 'next'

By clicking 'done', our device was set up.

Note: Take note of the organisation ID on the next page because it is needed later.

We needed to adjust the security settings for the device by clicking on the security tab on the left.

The policies page appears. We changed the connection Security, security level TLS optional and clicked 'save'.

That's created the device.

To use the Arduino IDE with this board, we first had to install the board, much like we did to be able to talk to the attiny chips, by pasting "<u>http://arduino.esp8266.com/stable/package_esp8266.com/index.json</u>" in *File > Preferences > Additional Board Manager URLs.*

	Preferences							
	Settings Network							
Sketchbook location:								
/Users/poppy-laptop/Docun	nents / Arduino	Browse						
Editor language:	System Default	(requires restart of Arduino)						
Editor font size:	10							
Interface scale:	Automatic 100 0% (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								
Aggressively cache compi	iled core							
Check for updates on star	tup							
🗹 Update sketch files to nev	v extension on save (.pde -> .ino)							
✓ Save when verifying or uploading								
Additional Boards Manager UR	Ls: c.json, http://arduino.esp8266.com/stable/package_esp8	3266com_index.json						
More preferences can be edite	d directly in the file							
/Users/poppy-laptop/Library	/Arduino15/preferences.txt							
(edit only when Arduino is not	running)							
		OK Cancel						

We copied the Sketch below into Arduino, scrolled down and filled in the Broker ID which is the Organisation ID that appears on the Device Drill Down Page.

Note: What to fill in for the Client ID is shown in the comments on the Sketch. That solves the connection to IBM.

Then we changed the WiFi connection in the same way.

Putting it All Together

Back to Node Red we double clicked on the the 'IoT out' node in our flow and used the credentials from our device.

When we hit deploy a little green square appeared below the node to show it's connected and we could see them on the dashboard by clicking the dashboard tab on the right hand side and the link icon - a little box with an arrow coming out of it.



Ś	Firefox File Edit View	v History Bookmar	ks Tools Window	Help				🚽 🚳 😨	⊕ \$? •	43% 🔳	Wed 21:08	< ≡
	🗢 🔿 💿 🧹 🏧 Mali –	🏒 High-L 🥯 Ardui	n 🙍 How tr 🔞 S	gn u, 🔯 Mail –	🐯 Sign in 🛛	😵 Sign in 🛛 🖄 Servic.	🜁 IBM W 🛛 😤 Node-	≫ No X	🛕 Proble 🛛 🗟 N	ode- >	+ ~	
	← → ♂ ☆	🛈 🔒 https:	//testing3.eu-gb.myt	luemix.net/ui/#/0				… ◙ ☆		± III\	9 🗉 🗉	
	🛱 Most Visited 🍯 Raspberry F	i • View 🔘 3D Scanner	Fab Academy Assig	🖨 Fab Classes	😚 My FabAcad	demy Site						en Shot
	Final Result											
			Servo 🔵			Default						en Shot
												3012//080#3
~												
-												en Shot
						LED Colour	0					621.00.18
												en Shot 620.58.12
						Stepper Steps		-				
												erplate
												en Shot
												50.49.33
												en Shot
and P												60.35.34
									a lana lina	-		
		I 🛞 🛃 🚭		r 🜔 🧕	9			e 🖄				

Arduino Sketch

/*

* Copyright 2018 IBM Tim Minter

*

- * Licensed under the Apache License, Version 2.0 (the "License");
- * you may not use this file except in compliance with the License.
- * You may obtain a copy of the License at

*

* apache.org/licenses/LICEN...

*

- * Unless required by applicable law or agreed to in writing, software
- * distributed under the License is distributed on an "AS IS" BASIS,
- * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

* See the License for the specific language governing permissions and

* limitations under the License.

*/

// You will need to install all the libraries listed in the includes section via the Arduino Library Manager.

// If using the setup_wifiHotSpot option....

// If the board is started, and a known wifi connection is available, it will connect automatically.

// If no known connection is found the board will switch to access point mode and a wifi network will be created with the ACCESS_POINT_NAME set below.

// Using anotherdevice you cna the connect to that wifi connection using the ACCESS_POINT_PASSWORD set below.

// Your browser will be opened to a configuration page where you can set up the wifi credentials for this board.

// This section is where you can configure all the things that need configuring

// wifi setup const char* WIFI_NETWORK_NAME = "eaglelabsToYou"; const char* WIFI_PASSWORD = "eagle2You";

// update this with the Broker address from IBM Watson IoT Platform const char* BROKER = "nzs2ny.messaging.internetofthings.ibmcloud.com"; // update this with the Client ID in the format d:{org_id}:{device_type}:{device_id} const char* CLIENTID = "d:nzs2ny:wemos:thing13"; // update this with the authentcation token const char* DEVICE_AUTH_TOKEN = "thingthing";

//LED (Neopixel) setup
#define LED_CONTROL_PIN D1
#define NUMBER_OF_LEDS_CONNECTED 3

//servo setup #define SERVO_CONTROL_PIN D2

//IoT and wifi libraries to include #include <ESP8266WiFi.h> #include <PubSubClient.h>

//LED (Neopixel) libray to include
#include <Adafruit_NeoPixel.h>

//servo control library to include #include <Servo.h>

//wifi includes

#include <DNSServer.h> //Local DNS Server used for redirecting all requests to the configuration portal #include <ESP8266WebServer.h> //Local WebServer used to serve the configuration portal #include <WiFiManager.h> //https://github.com/tzapu/WiFiManager WiFi Configuration Magic

//json crunchig library #include <ArduinoJson.h>

//define servo (the pins will be defined later)
Servo servo1; // create servo object to control a servo

// subscribe to this for commands: #define COMMAND_TOPIC "iot-2/cmd/command/fmt/text"

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMBER_OF_LEDS_CONNECTED, LED_CONTROL_PIN, NEO_GRB + NEO_KHZ800);

WiFiClient espClient;

PubSubClient client(espClient);

int servoPosition = 90;

void setup() { Serial.begin(9600); Serial.println("Hello world!"); // prints hello with ending line break strip.setBrightness(180); //max is255 strip.setPixelColor(0, strip.Color(255,0,0)); strip.show(); delay(500); Serial.println(); Serial.print("MAC: "); Serial.println(WiFi.macAddress()); servo1.attach(SERVO_CONTROL_PIN); // attaches the servo on pin D4 to the servo object strip.begin(); strip.show(); // Initialize all pixels to 'off' setup_wifi(); client.setServer(BROKER, 1883); client.setCallback(callback); strip.setPixelColor(0, strip.Color(0,0,0)); strip.show(); } void setup_wifi() { // Start by connecting to the WiFi network Serial.println(); Serial.print("Connecting to ");

Serial.println(WIFI_NETWORK_NAME);

WiFi.mode(WIFI_STA);

WiFi.begin(WIFI_NETWORK_NAME, WIFI_PASSWORD);

wait_for_wifi();

```
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
```

```
}
```

}

}

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.println("Message Received");
  const size_t bufferSize = JSON_OBJECT_SIZE(5) + 30;
  DynamicJsonBuffer jsonBuffer(bufferSize);
  const char* json = (char*)payload;
  JsonObject& root = jsonBuffer.parseObject(json);
```

```
int LEDAddress = root["A"];
int R = root["R"];
int G = root["G"];
int B = root["B"];
```

```
if(LEDAddress && R && G && B){
    //change the colour of the specified LED (if the colut and position has been sent
    strip.setPixelColor(LEDAddress, strip.Color(R,G,B));
    strip.show();
} else if (R && G && B){
    //change the colour of all LEDs in any attached strip if position has not been sent
    for(uint16_t i=0; i<strip.numPixels(); i++) {
        strip.setPixelColor(i, strip.Color(R,G,B));
        strip.show();
    }
}
if(root["servoPosition"]){//only if a value for this has been sent take action otherwise don't set the position
        servoPosition = root["servoPosition"];</pre>
```

Serial.println(servoPosition);

}

```
void wait_for_wifi(){
 Serial.println("waiting for Wifi");
 while (WiFi.status() != WL_CONNECTED) {
       delay(1000);
       strip.setPixelColor(1, strip.Color(255,0,0));
       strip.show();
       delay(500);
       strip.setPixelColor(1, strip.Color(0,0,0));
       strip.show();
 }
 Serial.println("");
 Serial.println("WiFi connected");
}
void reconnect() {
 boolean first = true;
// Loop until we're reconnected to the broker
 while (!client.connected()) {
       if (WiFi.status() != WL_CONNECTED) {
       wait_for_wifi();
       first = true;
       }
       Serial.print("Attempting MQTT connection...");
       if (first) {
       first = false;
       }
       // Attempt to connect
       if (client.connect(CLIENTID, "use-token-auth", DEVICE AUTH TOKEN)) {
```

```
Serial.println("connected");
       strip.setPixelColor(1, strip.Color(0,0,255));
       strip.show();
       }
       else {
       Serial.print("failed, rc=");
       Serial.println(client.state());
       strip.setPixelColor(1, strip.Color(0,255,0));
       strip.show();
       delay(500);
       strip.setPixelColor(1, strip.Color(0,0,0));
       strip.show();
       delay(500);
       strip.setPixelColor(1, strip.Color(0,255,0));
       strip.show();
       }
 // subscribe to the command topic
 client.subscribe(COMMAND_TOPIC);
void loop() {
 if (!client.connected()) {
       reconnect();
 // service the MQTT (IoT) client
 client.loop();
```

}

}

}

stepper.run();

```
//move the servo
servo1.write(servoPosition); //servo 1
}
```

Thanks to Tim Minter from IBM and Jon Paterson at Eagle Lab Brighton for helping us get to grips with Node Red

Grasshopper and Firefly

Grasshopper is a plugin for Rhino which enables parametric modelling. It is similar to Node-RED in that it has 'nodes' of code which alter data which is then passed onto subsequent nodes. It is open source which has allowed developers to create additional plugins. One very interesting one is called <u>Firefly</u> which allows Grasshopper to communicate with an Arduino getting information from its inputs and sending data back to control the outputs.

To get this to work you first need Rhino and Grasshopper installed on your computer and then download and install the <u>Firefly</u> plugin for Grasshopper. Next you need to upload the Firefly Firmata to your Arduino board. In the .zip folder downloaded from the FoodforRhino website open up the Firefly_Firmata Arduino sketch.



At the start is some information about the boards that are compatible and how the pins are used. It is important to open up this code in an older version of Arduino (Arduino 1.0.5) as otherwise there will be an error compiling due to the 'WritetoDAX' error. It is possible to <u>fix this</u>, but requires some editing of the Firmata





Once we had successfully uploaded the Firmata to the Arduino we can move into Grasshopper.



Above left we can see the full sketch we used to connect the Arduino to Grasshopper. It starts by getting the port 'available ports' from our machine. We then select which port the Arduino is connected to and run this through the 'open port' node. Here we can open and close the port and also set the baud rate (115200 for this example).

The right image shows the Arduino 'Uno Read' which gets the data from the pins of our arduino, here we can start and stop reading. It is also important to set the timer as this is the interval at which data is received.

With this sketch we are able to read the value coming from our LED (used as an LDR) which can be seen in the video below.

https://youtu.be/RScf1e6t14E

Python

Python is a general-purpose interpreted programming language. The written code is not actually translated to a computer-readable format at runtime, where most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

We wanted to use Python to create a graph based on data sent through the serial communication protocol.

To install and run Python programmes, you run a series of commands in the terminal window. The first one here installs XCode which has a set of command line tools. xcode-select --install

Then we install Homebrew, which is a package manager, here's the code to install it: /usr/bin/ruby -e "\$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" Then to install Python 2.7: brew install python@2

Homebrew installs Setuptools and pip for you. Setuptools enables you to download and install any compliant Python software over a network (usually the Internet) with a single command (easy_install). It also enables you to add this network installation capability to your own Python software with very little work. Pip is a tool for easily installing and managing Python packages. pip2 -V # pip pointing to the Homebrew installed Python 2 interpreter

You also need to install all the necessary libraries, including these (separately):

sudo pip install matplotlib sudo pip install numpy sudo pip install pyserial sudo pip install drawnow

We found another <u>youtube video</u> which seemed like a nice simple programme to do what we wanted in Python2.7, so we thought I'd give this a try, since the last few attempts have not gone well...and it works!!! Here's a screengrab of the raw analog data (noting that the data is actually mV, not temperature as suggested in the y-axis title...that will come later once the thermistor has been calibrated):



After installing Python2.7, we needed to open the corresponding Idle2.7. To do this: in the command terminal type: Idle2.7

This opens the python shell. Then file > open the .py file. Then to run it, goto RUN menu > RUN MODULE. Alternatively, in the command terminal, goto the python directory and type:

cd python

python2.7 filename.py e.g. python2.7 plot-serial-v4-WORKING.py

Here's the code for the arduino: int value; void setup() { // initialize the serial communication: Serial.begin(9600);

void loop() {
 // send the value of analog input 5:
 value=analogRead(A5);
 Serial.println(value);
 delay(500);

And here's the code to run in Python 2.7. import serial import numpy import matplotlib.pyplot as plt from drawnow import * values = []

plt.ion() cnt=0

serialArduino = serial.Serial('/dev/cu.usbmodem1421', 9600)

def plotValues():

plt.title('Serial value from Arduino') plt.grid(True) plt.ylabel('Temperature Deg C') plt.plot(values, 'rx-',label='values') plt.legend(loc='upper right')

for i in range (0,600):

values.append(0)

while True:

while (serialArduino.inWaiting()==0): pass valueRead = serialArduino.readline()

try:

valueInInt=int(valueRead) print (valueInInt) if valueInInt <= 1024: if valueInInt>=0: values.append(valueInInt) values.pop(0) drawnow(plotValues) else: print "Invalid! negative number" else: print "invalid too large" except ValueError: print "Invalid! cannot cast"

and here are the files to download:

• plot-serial-in-python.ino

• plot-serial-in-python.py

here's a video of me loading idle2.7 and python2.7, opening the file, and running the graph: <u>https://youtu.be/Y1Ufu_ieVwc</u>