Post: Assignment 0 - Introduction

October 17, 2024 1:50 PM

My introduction post:

https://landing.athabascau.ca/blog/view/25195480/assignment-0-introduction

Comment: Jon Newman: Assignment 0: Arduino and Orientation

October 17, 2024 1:52 PM

I responded to a post:
https://landing.athabascau.ca/blog/view/25134536/assignment-0-arduino-and-orientation

With:

Thanks for your post Jon! I share a lot of your same experience with previous Arduino IDE installation and messing around - they're great hobby devices! I wouldn't worry too much about the messy code though, from what I read in the Instructors weblog that c+ programming isn't too integral to the course, a lot of it is functions calls and procedural coding.

Good luck with your studies, and thanks for sharing.

B.t.w. Did you have a horrible time posting to Landing? It's taken me a few tries to upload my first post.

From <https://landing.athabascau.ca/blog/view/25134536/assignment-0-arduino-and-orientation>

Post: Unit 0 : Experiment: Blinking LED and SOS

October 17, 2024 1:59 PM

Link to post about blinking LED and SOS patterns:
https://landing.athabascau.ca/blog/view/25196753/unit-0-experiment-blinking-led-and-sos

Post: Unit 1: Weblog and Exercises

October 17, 2024 2:00 PM

Poste about Unit 1 Exercises:
https://landing.athabascau.ca/blog/view/25296772/unit-1-weblog-and-excercises

Reply: Oluwatomi Fadiora: Circuit 4b

October 27, 2024 10:39 PM

I responded to this post:
https://landing.athabascau.ca/discussion/view/25414705/circuit-4b

With:

Hello! I noticed my temperature readings were definitely elevated above room temperature, however not 55 degree high! I believe the max I saw was 30-35, a good 7 or 8 degrees above my room temperature.
(Winter's coming !)
I recall reading in SIK:
"Make sure that you wired the temperature sensor correctly. The temperature sensor can get warm to the touch if it is wired incorrectly. You would need to disconnect your microcontroller, rewire the circuit, connect it back to your computer, and open the serial monitor."

Sorry for the late reply, and GOOD LUCK!

IDE1: Integrated Development Environment Options? (part 1)

October 14, 2024 2:47 PM

Background:
While preparing for this course I noticed various alternative options for Arduino IDE and I'm curious if anyone has tried any of them?

Some of the options I found were:
- Arduino IDE for VisualStudio
- Arduino Pro IDE,
- VS Code Marketplace Enhancement
- Emacs
- PlatformIO
- Etc - leave a comment if I missed one!

I've been developing professionally Microsoft's IDE for a very long. The first version I used, was a barely functional Visual Studio 6.0, and have used practically every version since. I haven't used VS Code very much, as I'm usually in Pro 2024, so I'm really interested in doing a review of the Arduino IDE for Microsoft Visual Studio 2024.

Some criteria I'm going to look into:
- Ease / comfort of use
  - Easy to setup? User friendly? Are there keyboard shortcuts? Is it documented well?
- Available features
  - Intellisense, Source Control, Profilers
- RedBoard integration
  - Can I use it for this course?

Preface:
Git integration

My primary reason for this review is my desire to use Git. I find it useful for protection codes, especially while developing for more complex systems, as well as provide an easy way to share my work with instructors.
I was hoping for an all in one solution with Arduino IDE but none was available. Instead I've opted to use Git Desktop and folder based integration - it works fine, but takes too many steps. Sometimes I jump into Command Prompt and use the CLI, but not very often.

Installation
Installation was very easy, and free! I used the Windows Marketplace to install VS Code, took almost no time. Inside VS Code Extensions, there were dozens of Arduino options available! Many options were available with a range of features. Strangely enough, the most downloaded option "Arduino" by "moozyk" seemed to be the worst option, and lowest rated. I opted for "Arduino Community Edition" as it was very well rated, and came complete with Serial port monitor, intellisense, syntax highlighting, board and library manager. Very impressive!

Arduino Community Edition for VS Code - First Impressions

Opening an existing project:
I started easy, with my first challenge - the blinking light. I opened it by selecting the folder. Interestingly enough, VS Code immediately prompted me with an intimidating window, asking if I trusted the author. I was not expecting that, luckily, I am the author so I said "yes". Voila the ino file was opened, with beautiful colour syntaxing. A little less monotone as Arduino IDE, so far so good. Uhoh, another warning, this time I had to install c++ extensions - no problem, VS Code did all the work. A minute later, and I'm back to my ino file - with 13 problems!

It's at this point, that I'm thinking to myself, I haven't specified what I'm coding for, there must be missing libraries and configurations at this point.

And it's at that point that everything came to a halt.

The documentation I found said I'd have to install libraries, they were missing. I had to configure the board, but that touted 'Board Manager' wouldn't open. Similarly, the 'Board Library' failed to open as well. No errors messages, just nothing happened.

It is at this point that I learned the project is dead, and no longer maintained.
I wish I'd found this earlier, but a large thread on the Arduino site concludes this extension for VS CODE is a no go…

There is hope! A community driven group has resurrected a previous fork of the extension, and alive here:
https://github.com/vscode-arduino/vscode-arduino/issues/13
Maybe I should take a look at it after this course!

For now, I'm abandoning Arduino extension to try out **PlatformIO**! Stay tuned, that's coming next…

IDE2: Integrated Development Environment Options (part 2)

October 27, 2024 10:42 PM

Background

Install PlatformIO into VS Code

Finally blowing the dust off of part 2 PlatformIO review just in time to submit the final project. I had intended on attempting this as a replacement IDE for the final project, however I ultimately decided I had a steep enough learning curve handling all the projects tasks.

Here are my experiences…

Install was very straightforward, even without instructions. It took me a second to notice the PlatformIO icon on the left, but as soon as I clicked it, the remaining installation steps were completed. I was prompted to reload VS Code and voila, done!

Research

Before getting to far into things, I decided to watch a few tutorial videos on youtube. Listed below are a few of the videos I found helpful. Most importantly is the Platform.IO website itself.

https://docs.platformio.org/en/latest/integration/ide/pioide.html

As described the IDE promises a lot of improvements over the Arduino IDE, some features include:

- Cross-platform build system without external dependencies to the OS software:
  - 1,000+ Boards
  - 40+ Development Platforms
  - 20+ Frameworks
- Debugging
- Remote Development
- Unit Testing
- C/C++ Intelligent Code Completion
- C/C++ Smart Code Linter for rapid professional development
- Library Manager for the hundreds popular libraries
- Multi-projects workflow with multiple panes
- Themes support with dark and light colors
- Serial Port Monitor
- Built-in Terminal with PlatformIO Core (CLI) and CLI tool (pio, platformio)
- Built-in PlatformIO Home.

Build an Arduino project with PlatformIO IDE for VSCode

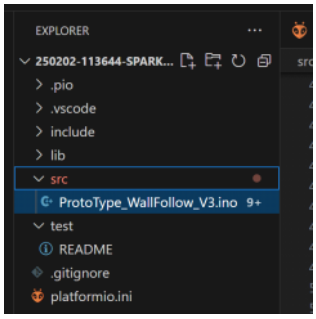
And the much more helpful and detailed:
PlatformIO: All you need to know in 10 Minutes!

Impressions

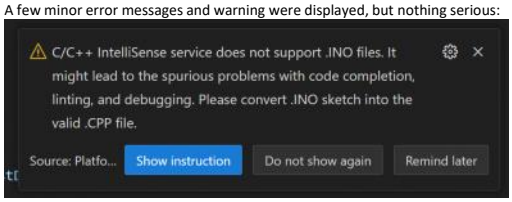
For this experiment, I'm going to import my final project from Arduino IDE, and restructure into a more organized and clean project structure that I'm accustomed to.

Review 1: Importing

The import process is quite simple, and a new project structure was created for me:

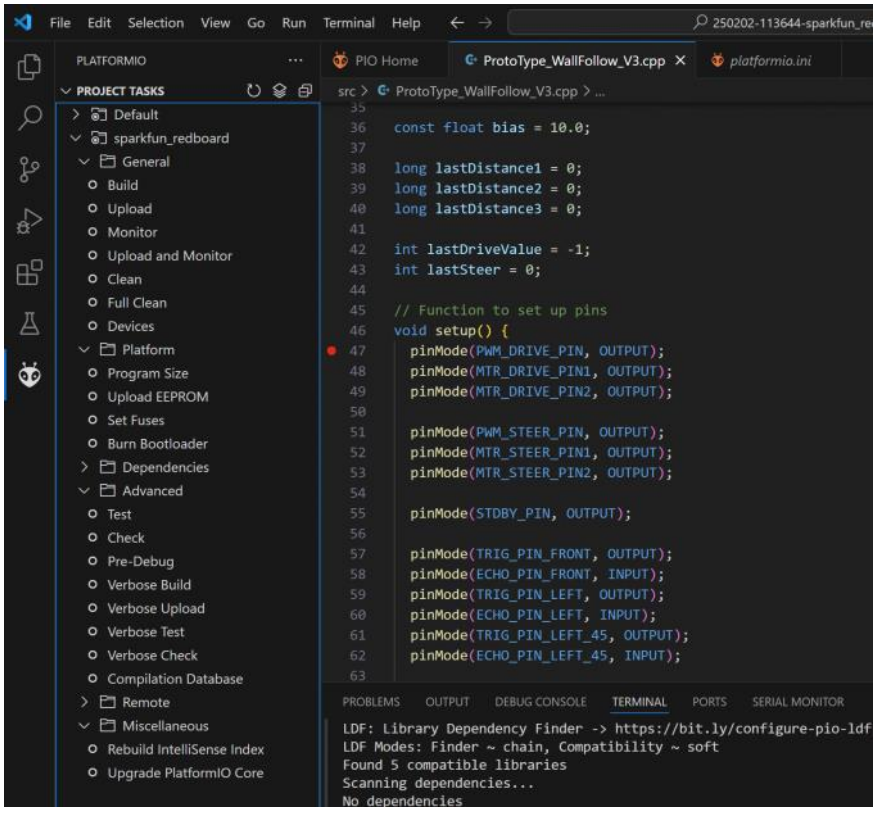


A few minor error messages and warning were displayed, but nothing serious:



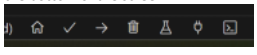
Apparently the .ino file type that Arduino IDE uses is not an official c++ file type, and not supportd by all IDEs, so it's recommended to convert from ino to cpp.
Here is the justification and how to here:
[Convert Arduino file to C++ manually — PlatformIO latest documentation]

Wow, that was easy! Conversion to the new file type was simple, and immediately the project compiled cleanly. Now that we've completed all of thise, let's have a look at the ide.



An initial review shows an impressive offer of functionality. The import process already created a nice project structure and even dependencies - something much more familiar to me.

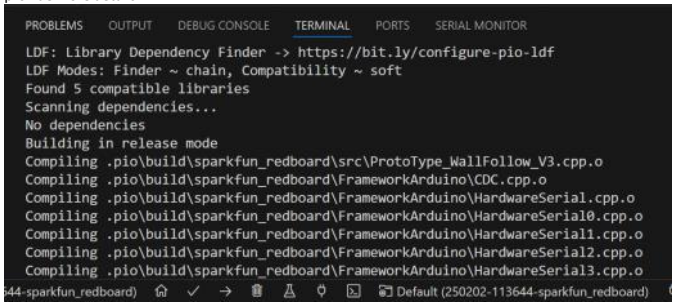
In addition to frequent tasks listed under project tasks there is a handy list of functions displayed at the bottom of the screen:



Review 2: Compile and Upload

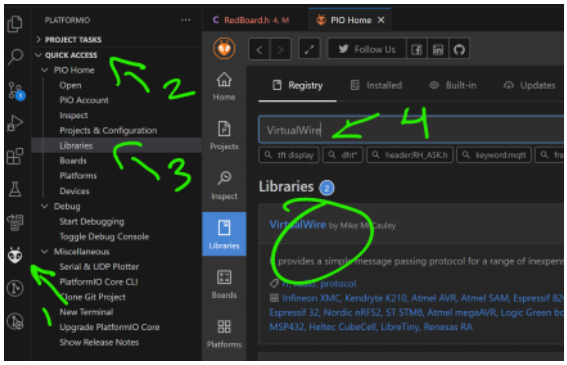
A very important process is the actual compiling and uploading of the code to the robot. This is a process run hundreds of times, it must be simple to do, with clear feedback on the process as well as code or transfer issues.

As mentioned earlier, compiling or uploading can be easily initiated with the quick tasks or the icons at the bottom of the IDE. The output is captured from a terminal window and outputed there - no real improvement on the Arduino IDE, however it appears a different compiler is used, which provide more details:



Review 3: General IDE comments

The library manager is also just as easy as the Arduino IDE, I was able to easily add the VirtualWire library (required for RF communication) by a few clicks:

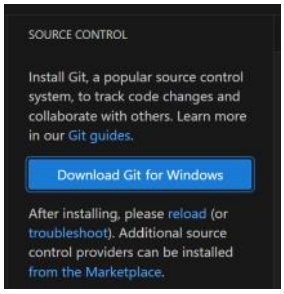


Since the PlatformIO is an extension that is built upon the VS Code software, any user with existing Visual Studio experience would greatly benefit from this IDE. It uses all existing features, short cut keys and community support. If you are comfortable with VS code or Visual Studio, I highly recommend this!

Review 4: Source Control

As mentioned previously, I'm a huge fan of source control and rely upon it heavily for my day today development tasks. PlatformIO and VS Code supports source control in many implementations, I prefer to use Git, which I have done so for my entire COMP 444 career.
Currently I have been using Git Desktop, a separate application than Arduino IDE. This meant that in addition to doing development work in the IDE, I would have to open a separate application to manage pushes, pulls and pull request management.

First step is to install the Git for Windows, and extensions:

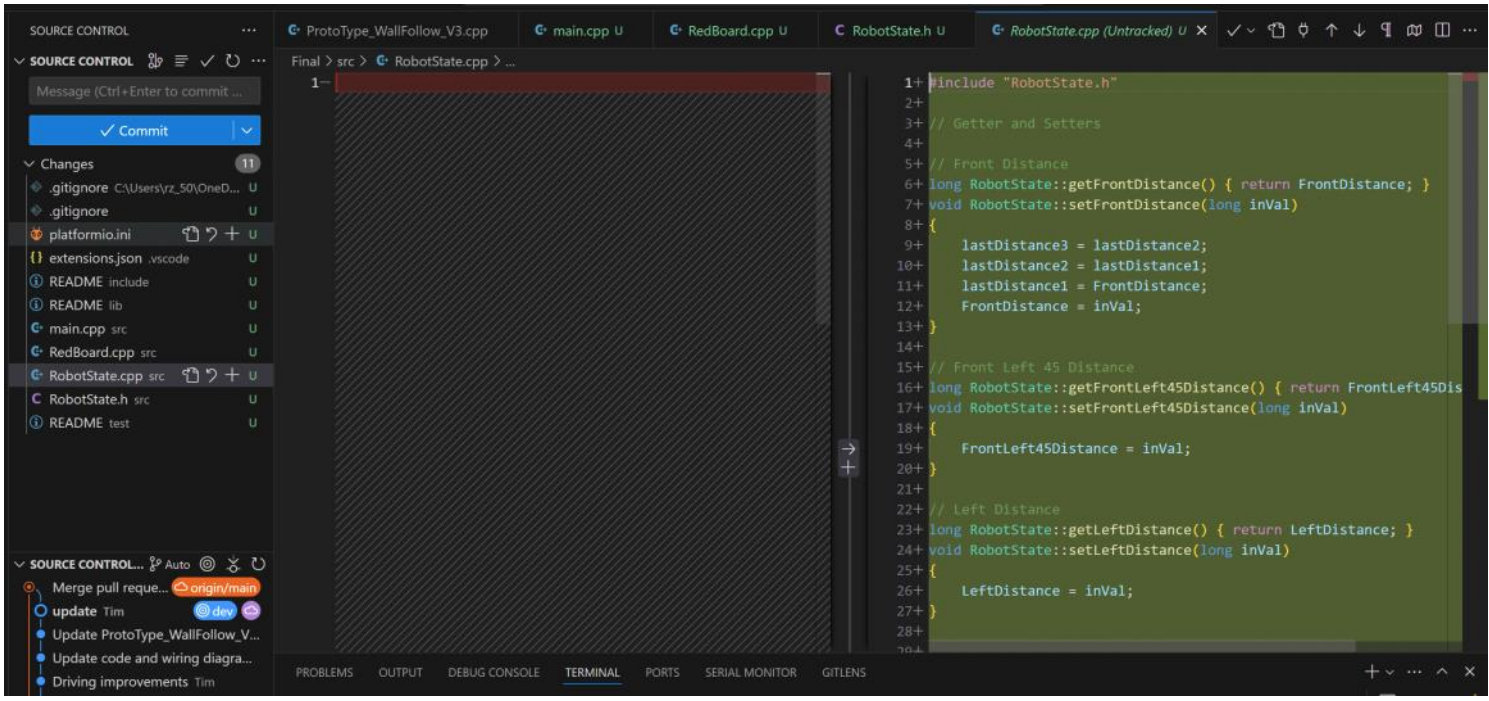


A few configuration prompts, and a restart of VS Code extensions later, and I'm away at the races. Upon selecting the root folder of the project PlatformIO detected my previous Git repository and automatically configured everything for me - what a great experience!

So far things are looking very good for VS Code and Platform IO. Let's make a few code changes and evaluate the push process.

(a few moments later)

I've made a few changes and am ready to review and push to source control. Naturally VS Code has built-in SourceControl interface that fits seamlessly into the VS Code IDE. A very easy and straightforward experience, and no more messing around with an external application!



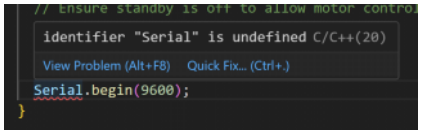
Review 5: Code Modifications and General Impressions

The final prototype of the wall follower project involved having all code within a single file. It made a very long, hard to read and disorganized source code file. I detest organized code, the file should be logically laid out with clear separation of regions of the file including variables, functions, related

functional areas etc. For this review, I will refactor my prototype code into a final multi-file structure.

The refactoring, reference checking of VS Code is just as good as Visual Studio, and something simply not offered in Arduino IDE, or insufficiently provided. A huge win for Platform IO.

While splitting code from a single large file, to separate smaller files, I missed reference includes. VS Code easily identified them as missing, before a compile was even performed. VS Code also provided 'Quick Fix' to include missing references - a feature I sorely missed from Arduino IDE.



The keyboard shortcuts are also much more capable than with Arduino IDE, and naturally follow the Microsoft prebuilt configurations.

Final Word

Wow, just amazing! PlatformIO is a fantastic IDE extension for VS Code and Arduino development. I've been able to refactor and recompile my prototype project into a presentable, organized project. The interface can be a little intimidating Arduino IDE simple, to the point interface provides all the necessary functionality for a beginner developer, however as I am much more accustomed to a full featured IDE, I really appreciate everything PlatformIO has to offer!

PlatformIO - a great advanced replacement for Arduino IDE!

Project Proposal (and follow up)

February 2, 2025 10:08 PM

Posted on: October 31 2024
https://landing.athabascau.ca/discussion/view/25610347/project-proposal-homeassistant-bot

Proposal
An autonomous bot that will drive around the main floor of my house, entertaining my cats, detecting light levels, and interacting with my HomeAutomation system using radio frequency.

By this, I hope to demonstrate wireless data receiving and transmission, proportional motor control, light detection, advanced driving with collision avoidance through use of multiple distance and collision sensors.

Background

I'm a avid home automation enthusiast. I've automated nearly every aspect of my house (from doors locks, garage doors, lights, cameras, furnace/air conditioners, radiant floor, Roomba, gas/water consumption and more). Nearly 1,400 data points are tracked or managed by the system.

There are a few things that my system is unable to accomplish, that I believe a robot could assist with. The first problem is, my cats tend to get very bored, and when they're bored they fight with each other. That always leads to tufts of hair on the floor, and sometimes occasional minor injuries. The second problem is occasionally lights get left on over night.

I would like my robot to drive around the main level of my house, for 5 minutes at a time, avoid obstacles and falls, sense the ambient light levels and beam a cat laser pointer toy on the floor. Based on the incredible improvements I noted during Coding Challenge of Circuit 5C: Autonomous Robot, I intend on using a proportional response to obstacles. The robot will remain in a standby mode until commanded to operate. When the command is received, the robot will start its 5 minute circuit. If light is detected the home automation server will be notified.

The main body of the robot will be very similar to the SIK exercises, however I will be extending the breadboard to allow for additional sensors. The robot will be enhanced with multiple distance sensors to enhance its obstacle detection, including a downward facing sensor to detect and prevent falls. The robot will also use a 433 MHz Radio Frequency transmitter and receiver. These components will allow the robot to communicate wirelessly with my home automation server. I have already purchased these components.

Responsibilities
The robot will:

- Follow a random circuit around the house, prioritizing the straightest trajectory
- Create directional algorithm to avoid circles, by alternating left / right correction behaviour upon hitting obstacles
- If any obstacles are encountered during the track, the robot will follow an avoidance procedure (as demonstrated in Coding Challenge of Circuit 5C Autonomous Robot); leave the track, avoid the obstacle, then join the track.
  Use a photo-sensor to determine which lights are left on in a room. Entertain the cats with a laser toy

- Mount a cat laser toy to the robots body, pointing to the ground
  The robot will transmit a data via 433 MHZ RF upon light detection

- RF is very low battery usage, and easy to use
- My current home automation server has several RTL-SDR radios including one monitoring 433MHZ spectrum, decodes data, triggering automation scene: "Main Level Lights Off".
  Store running time in internal state, and after X hours of running transmit charge battery message

- Transmit using 433MHZ
- X (number of hours) will be determined during development / testing

Environment Requirements

The robot will require certain aspects of my environment be met:

- The circuit around the house must be completely navigable, minor obstacles are acceptable
- If the robot hits too many obstacles within a certain distance the robot will error out
  The home automation sever is running

Components / Sensors:

- Distance sensor (x4); Forward, Reverse, 30-45 degree Left / Right
- Motors

- RF transmitter
- Photosensor
- Battery
- RedBoard
- Robot body
- Cat Laser toy
  Cost:

- Additional components are needed to meet requirements of this robot
- 433 MHZ Transmitter/Receiver Kit: (6 pcs: 3 RX, 3 TX): $14.70 (already purchased)
- Distance Sensors (5 pcs): $20.99 (already purchased)
  Free / Already on hand:

- Home automation server (HomeAssistant Docker on RaspberryPi)
- Cat laser pointer
- Double sized breadboard

Replies

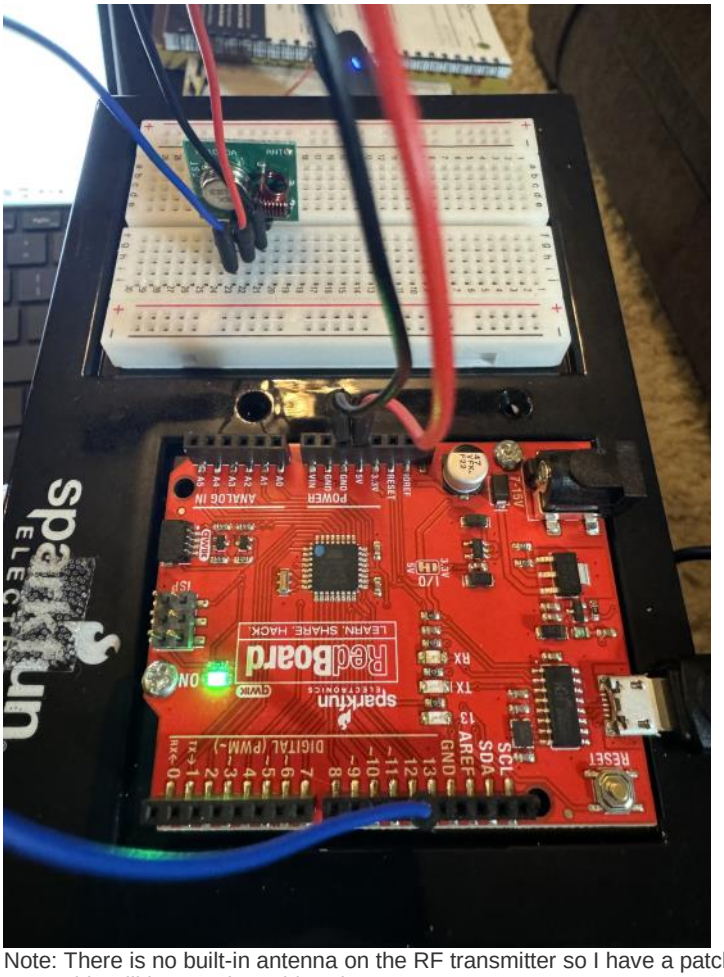
-
- Edit
  Timothy Blake *November 10, 2024 - 7:51pm*
  Update #1: Radio Frequency Proof of Concept
  I want to prove that I'm able to work with the RF transmitter and send data to the HomeAssistant server.

  Starting off was a little slow, this was probably one of the first Arduino circuits I created from scratch, most other exercises involved editing already existing sketches.

  I found a Arduino library called VirtualWire which provides all the functions I needed to communicate with the radio.
  Within a few minutes I had the breadboard setup, with the RF transmitter. The data was sent, and my server picked it up.

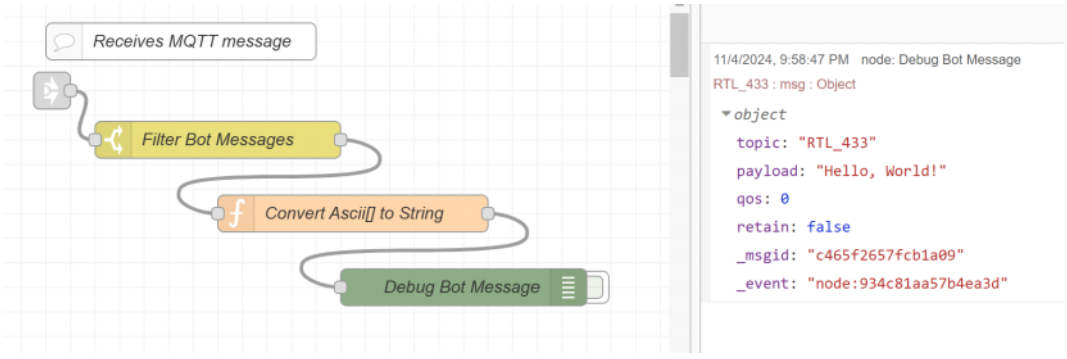
  Here is the source code of the project:
  https://github.com/malcolmsdad/COMP444/tree/main/Project/Investigations/RF_ProofOfConcept

  And the image of the board:



Note: There is no built-in antenna on the RF transmitter so I have a patch cord temporarily acting as one. This will have to be soldered on.

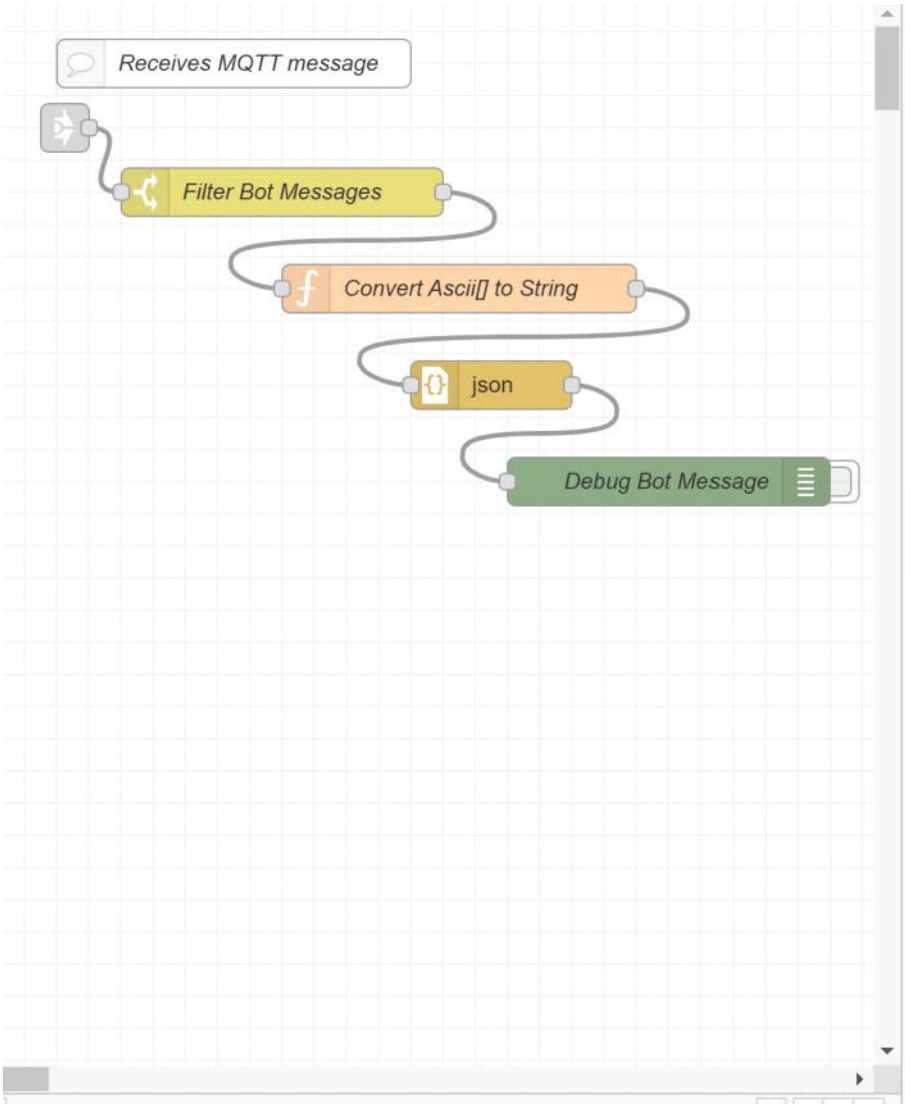
Here's the NodeRed Flow and the debug output of the message:



Next step is to include meaningful information, so I'd like to add a light sensor to the board, read from that devices, then transmit the light sensor value with in the message.

That also meant that I'd need to defined a data packet structure. That means to standardize and establish a contract of how the robot and server will communicate. My server prefers using JSON data structures, so that is what the robot will transmit. This will be the first draft of that package:

{
"Lux":XXX
}




```
_event: "node:934c81aa57b4ea3d"
11/4/2024, 10:38:01 PM node: Debug Bot Message
RTL_433 : msg : Object
object
topic: "RTL_433"
payload: object
lux: 947
qos: 0
retain: false
_msgid: "029229b07595eea5"
_event: "node:934c81aa57b4ea3d"
11/4/2024, 10:38:03 PM node: Debug Bot Message
RTL_433 : msg : Object
object
topic: "RTL_433"
payload: object
lux: 959
qos: 0
retain: false
_msgid: "7a3b76cf797d68c2"
_event: "node:934c81aa57b4ea3d"
11/4/2024, 10:38:05 PM node: Debug Bot Message
RTL_433 : msg : Object
object
topic: "RTL_433"
payload: object
lux: 973
```


Here is the working code:
https://github.com/malcolmsdad/COMP444/tree/main/Project/Investigations/RF_LuxDataPacket

And the NodeRed debug information, notice the extra step to decode JSON object. I also tested the sensor by blocking light to it, note the change of values (highlighted)
So far so good!!!