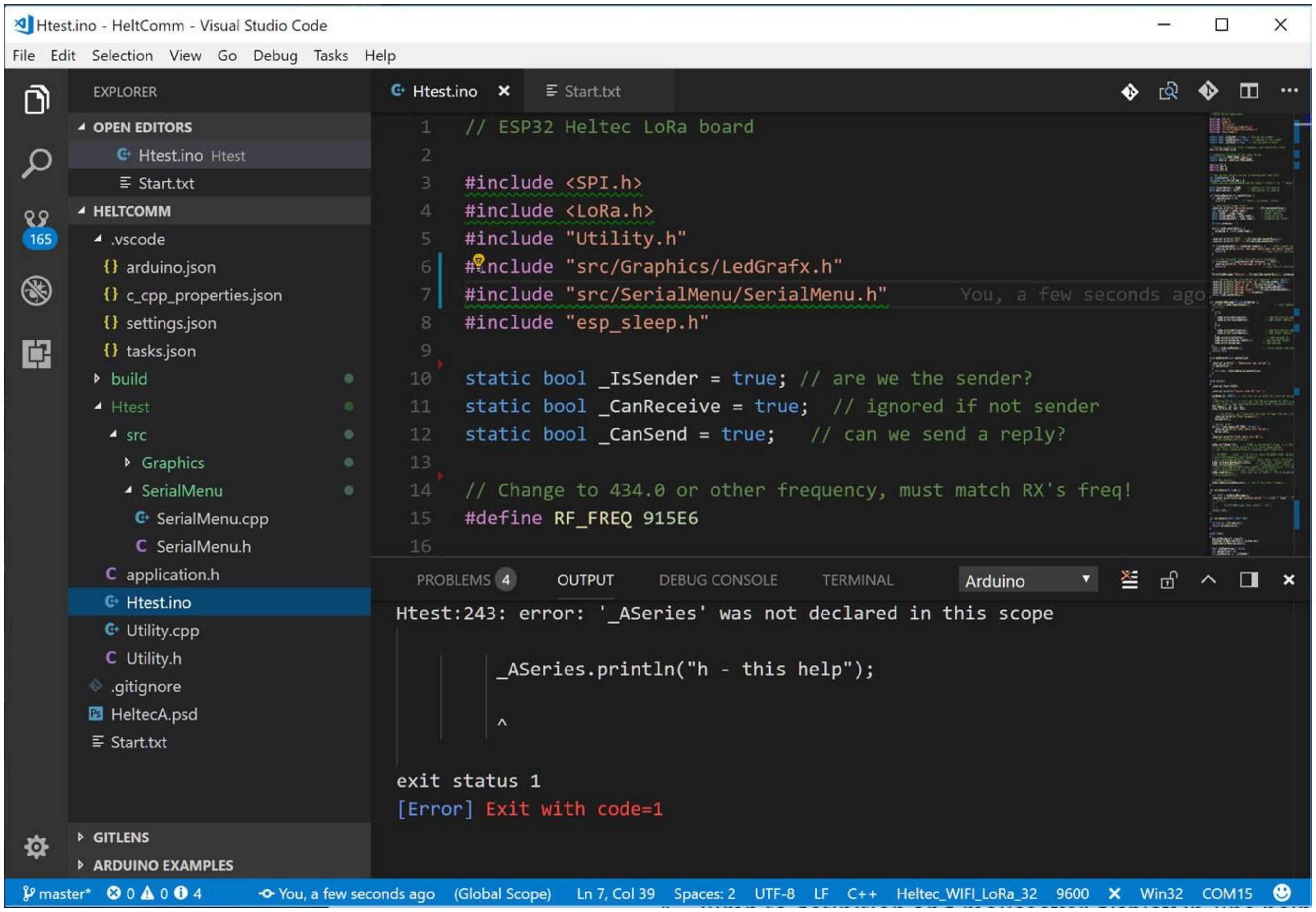
Use Visual Studio Code for Arduino

Mark Zachmann

Feb 19, 2018

VSCode has an Arduino plugin available in preview. Use it. Put your Arduino desktop application in cold-storage. Comparing the two is like comparing a beaver with a 2 ton backhoe.



```
// ESP32 Heltec LoRa board

#include <SPI.h>
#include <LoRa.h>
#include "Utility.h"
#include "src/Graphics/LedGrafx.h"
#include "src/SerialMenu/SerialMenu.h"          You, a few seconds ago
#include "esp_sleep.h"

static bool _IsSender = true; // are we the sender?
static bool _CanReceive = true;  // ignored if not sender
static bool _CanSend = true;    // can we send a reply?

// Change to 434.0 or other frequency, must match RX's freq!
#define RF_FREQ 915E6
```

```
Htest:243: error: '_ASeries' was not declared in this scope

         _ASeries.println("h - this help");

         ^

exit status 1
[Error] Exit with code=1
```

Visual Studio Code with an Arduino App

VSCode Advantages

1. Lots of useful key-stroke sequences, fully programmable.
2. Integrated with Git.
3. Intellisense that works.
4. Jump to definition and mouseover display in-line help.

5. Hugely useful on-screen click objects.
6. Full support for folder trees.
7. Plugin support.
8. Seriously have you used the Arduino IDE? Who needs a list?

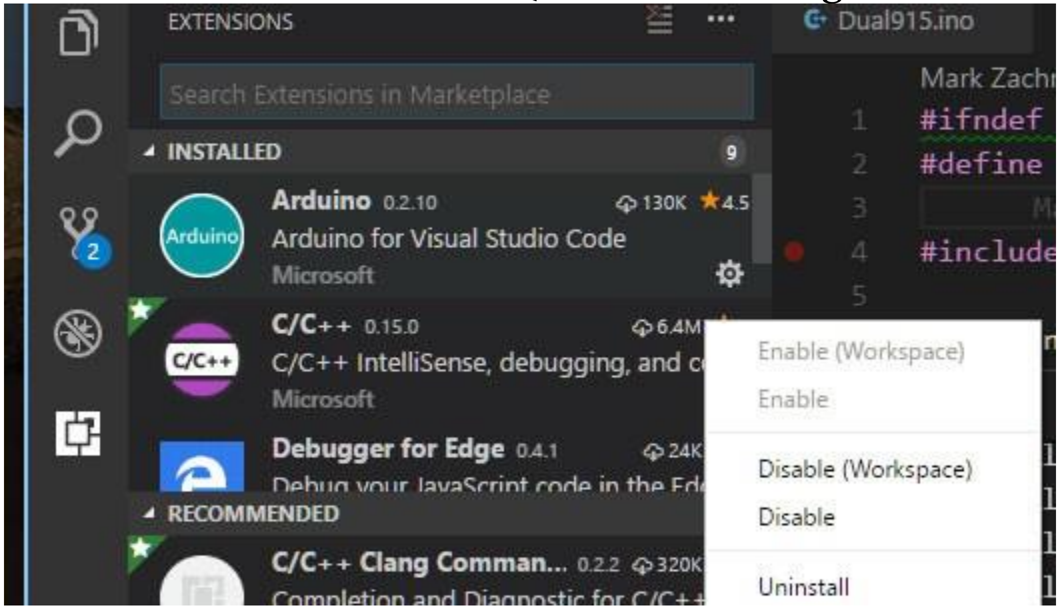
VSCode Disadvantages

1. For now you can't type into the serial monitor window. That's not cool.—*Update*: the latest version of the Arduino plugin now lets you send a string to the serial monitor. It's clumsier than a modem program but very usable.

Getting Started

Prerequisites

- A working version of Arduino desktop that can build.
- Visual Studio Code with Microsoft's Arduino Extension installed. Install it from the Extensions tab (below the debug button on the far left).


EXTENSIONS
Dual915.ino
Search Extensions in Marketplace
INSTALLED
9
Arduino 0.2.10
130K 4.5
Arduino for Visual Studio Code
Microsoft
C/C++ 0.15.0
6.4M
C/C++ IntelliSense, debugging, and c
Microsoft
Debugger for Edge 0.4.1
24K
Debug your JavaScript code in the Ed
RECOMMENDED
C/C++ Clang Comman... 0.2.2 320K
Completion and Diagnostic for C/C++
Enable (Workspace)
Enable
Disable (Workspace)
Disable
Uninstall

```
Mark Zachr
#ifndef
#define

#include
```

The Arduino Extension (per-workspace enabled)

Process

Assume you have a working project you want to *convert* to VSCode. Just

1. Run VSCode and **open** the project folder.
2. (optional but recommended) Use the source control tab to initialize a new **git** repository and check in your existing code.
3. From the command palette run: **Arduino: Initialize**
4. **Edit the project files** (see below) for any workspace-specific project variables (such as includes).

Taking Advantage of Visual Studio Code with Arduino

Once the Arduino extension is enabled for your project there are some tricks to make life better.

Use a src folder to hold your code folders

Strangely, arduino doesn't support subfolders (it won't automatically build and link them) unless they begin with a *src* subfolder root. See here: Recursive src Folder Discussion

Take a look at the picture at the start and make sure your source tree looks like:

```
ArduinoRootFolder
--- src
------- FolderA
------- FolderB
-------------SubfolderA
```

Add an 'output' option to arduino.json

Depending on how the plugin works today, you may not get an output option defined in the arduino.json settings file. Add this line just above the closing brace:

```
,"output": "build"
```

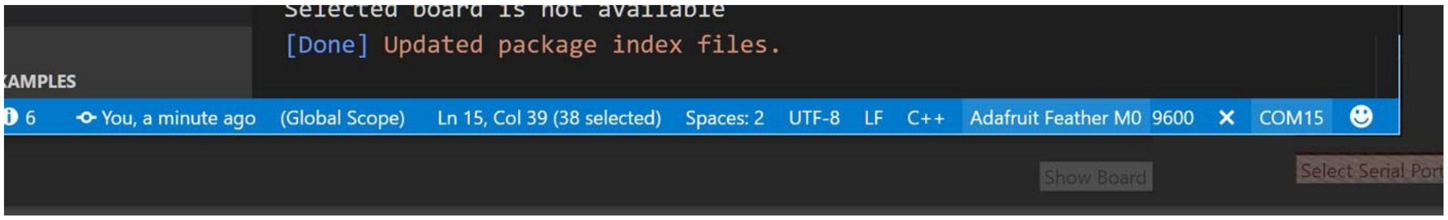
Here I've defined the *output* folder as *build*. It will create a build folder in the project root where the caching of all the Arduino stuff takes place. That has two advantages:

- It will decrease build time substantially. Strangely if you don't set output you don't get caching.
- Since Arduino is kind of bad at figuring out caches you can just delete the entire build folder contents to force a clean build.

Warning: from what I can tell putting the build folder inside your project tree sometimes causes issues. This is documented in the Arduino source. A better solution is to build in a folder outside of the project.

Investigate the buttons at the bottom

In typical new-age dumb software fashion there are lots of useful buttons on the bottom of the display that aren't obvious but that you should totally learn to use.



```
Selected board is not available
[Done] Updated package index files.
```

The bottom stripe on Visual Studio Code showing two mouse-over tooltips.

Mouse over them and you'll see all sorts of easy ways to change stuff.