

**Tech Studio Project**

Using

Slack API, Involt & Arduino

By

Lauren Harvey

# Step #1 – Set Up Slack API

Slack API is a built-in service with Slack Messaging that allows users to create their own apps, integrations and bots to improve the Slack experience. For this project, I created a new webhook integration that would post the message “SOMEONE FIXED A BUG!” to a Slack channel called “bugbot”, written from a bot named “bugbot”.

## Process:

1. Visit <https://api.slack.com> and navigate to Custom Integrations.
2. Click on ‘New Webhook’ to add a new webhook.
3. Follow the instructions on the page to set up the parameters of the integration, i.e. which channel to post to, what the name/icon of the bot will be.
4. Make a note of the Webhook URL.
5. Save Settings.

## Stumbles, Breakthroughs & Victories:

On the Slack API site, it is clear that there are a lot of options for users to create their own programs and bots for Slack. However, between apps, integrations, webhooks, bots, etc. it’s very difficult to differentiate which aspects of the site do what. Overall it isn’t very user-friendly. That being said, once you do find what you are looking for, it’s a very simple process (compared to the rest of the project), as it’s just a matter of filling in the blanks for what you need.

# Step #2 – Write cURL Code That Submits Data to Slack

The Slack API site provides a Webhook URL that acts as a target for code to be sent to. In fact, right next to the Webhook URL form on the website it specifies for you to “Send your JSON payloads to this URL.”

The code for the json payload/cURL request can be seen on the next page:



This php code can be saved into its own php file and will eventually be uploaded to an online server. As you can see, this code uses cURL to send a JSON payload to the $url above which is the Webhook URL that Slack API provided in the previous step.

It also includes the most important characteristics of the bot/app, which are the icon, the message and the channel ($icon, $message and $room respectively).

This code also randomizes the icon to be one of 5 bug images. These aspects can be chosen within the Slack API site, but this allows for a faster, simpler method of changing the information.

## Stumbles, Breakthroughs & Victories:

Having never had to use either cURL or JSON before, it was very difficult to understand what the Slack API meant by “Send your JSON payloads to this URL”. The website didn’t have any documentation to help new users write their own code, and it didn’t have any example code either.

I ended up having a major breakthrough after installing the Postman app, which is run through Chrome. This app is specifically designed to send these kinds of POST requests, and all of my tests run through the app were successful. Luckily or unluckily, the Postman app gives the user access to the codes given, but not the entire code – only an empty frame. With this I was able to fill out information that I needed and was able to send a successful POST request just by opening the above code in a web browser.

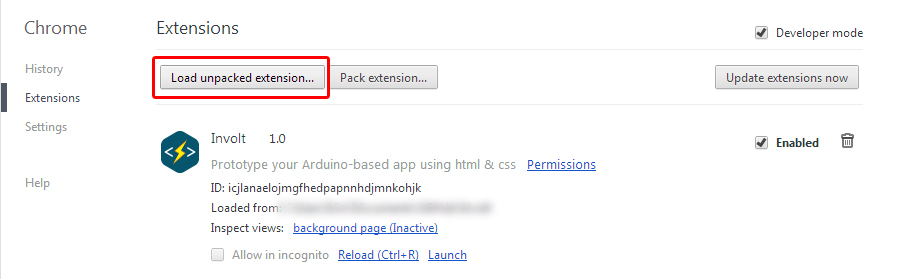
# Step #3 – Installing Involt

The next step of this project involved connecting an Arduino board to the Internet by giving it the ability to connect the serial port of the Arduino Uno. For this I used Involt, which is a developer extension using Chrome. This program is designed to connect the Arduino board to an HTML environment, so that you can design interfaces that interact with the Arduino board.

I didn’t necessarily need the HTML/CSS environment for this project, but this program was very helpful to me to bridge the gap between the Arduino and the computer. The installation of the app was reasonably simple, the process can be seen below.

## Installing Process (from <http://involt.github.io>):

1. Install Google Chrome.
2. Download and unpack Involt.
3. In Chrome go to tools > extensions.
4. Toggle Developer mode.
5. Click “Load unpacked extension…” and choose www folder.



## Stumbles, Breakthroughs & Victories:

This was by far the most difficult part of the entire project – figuring out how I could possibly connect the Arduino hardware with the computer to use on a web server. I initially didn’t even know that this step was important, but in hindsight it’s unfortunate that I overlooked it since I didn’t realize how difficult it would be.

I attempted many different ways of connecting the the Arduino via the serial port:

First, I tried a classmate’s suggestion to use Processing to connect to the Arduino. I managed to connect the two without too much trouble after writing code for both, but although I could get a digitalRead to show up on the Processing server I didn’t know where to go from there, so I had to move on.

Next, I tried a connection using an online reference code called PhpSerial.php which should theoretically open up the ports, check for input and keep checking, but after a lot of hard work I couldn’t figure out how to make the connection properly.

From there I tried an app called Breakout, but I had quite a lot of issues just installing the app onto my computer since it wouldn’t open. Eventually I determined that the app had to be run in the 32-bit version, even though my computer was 64-bit. After finally getting it to open properly, it was too complicated for my knowledge base and I had to move on again.

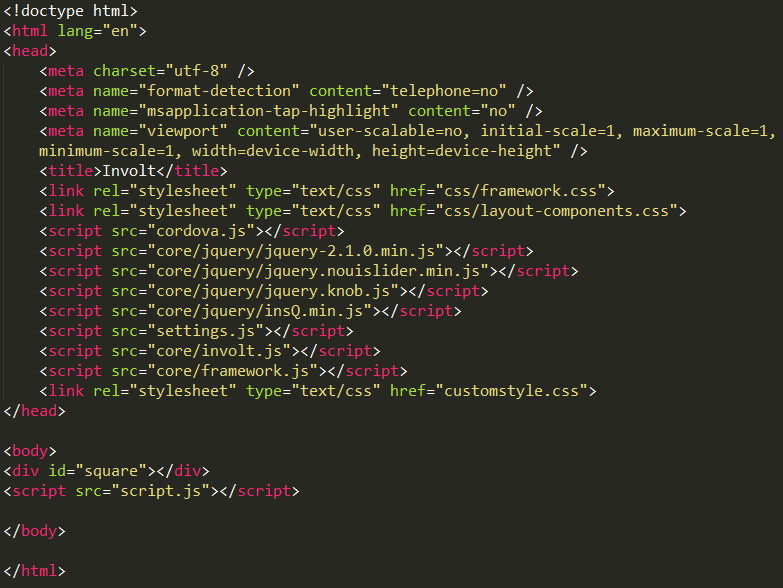
Finally, I attempted to use the Involt app and it was finally something that, although difficult to start using, wasn’t terribly difficult once you understood which scripts could be manipulated and which ones shouldn’t be touched. It took a while to figure out which was which, and I had to manually write a code in javascript to replace the program’s jquery, but it did manage to work.

My greatest victory in this step was when I was able to manipulate an example on the Involt website to suit my needs. The example on the site was an AnalogWrite script which I tested and got working, and I had to rewrite about 4 different script files to then change them to DigitalRead. I also wrote a javascript file with square colours to test the connections and functionality, which are still currently part of the interface (to show that the connections are still working).

# Step #4 Manipulate Involt index.html, javascript (Examples)

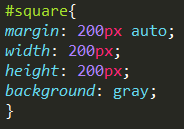
The Involt app, when you download and unpack it, includes over 50 files that all help to run the app and all of the scripts needed to connect to the Arduino. Of these 50 files, index.html is the code needed to run the main page of the interface.

The code for this can be seen on the next page:

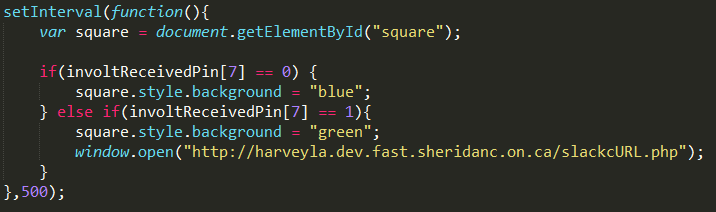


As you can see, this code references many of the other scripts in the Involt folder.

The id=”square” adds a square to the interface for testing purposes. The css for the square is seen below, and loads as gray upon loading.



The script.js was written by me to manipulate the square and, when the Arduino laser detector is hit by the laser, to run (open in a new window) the php code needed to post to slack. This code is seen below:



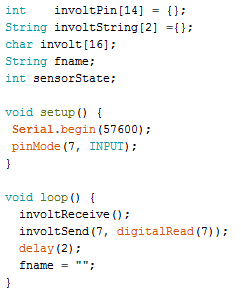
This changes the colour of the square to blue when the program is ready to accept new information from the Arduino (it doesn’t change to blue until it gets a signal from the Arduino, in this case that it’s not being hit). Once the laser detector is hit, the script changes the colour of the square to green and opens a browser window that automatically sends a POST request to Slack and posts a message to the bugbot channel!

## Stumbles, Breakthroughs & Victories:

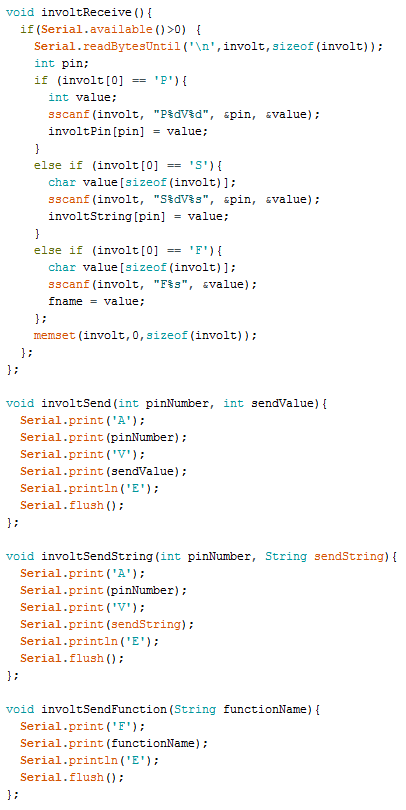
Connecting Involt to the web browser by using the window.open function isn’t the best, most efficient way of doing so, but it’s definitely the simplest. By using this strategy I can bypass the need to use ajax to connect my javascript function (client-side) to a php code (server-side). I consider this to be a victory, as (at the time) this was the final link to connect all aspects of my project.

# Step #5 – Manipulate Involt Arduino IDE Code

Once all of the software aspects are completed, the hardware aspect also has to be written and properly uploaded. Like above, Involt provided a sample Arduino Sketch file and, by manipulating it, I was able to get the Arduino to send a signal from the laser detector (whether a laser is hitting it or not) to the computer and register it as either a 1 or 0. Using this 1 or 0 I could then input that code into the Involt app info in the previous step and use it to manipulate the Involt UI. Below is the first half of the code for the Arduino Sketch IDE that I wrote myself:



Below is the second half of the code for the Arduino Sketch IDE that was provided by Involt:



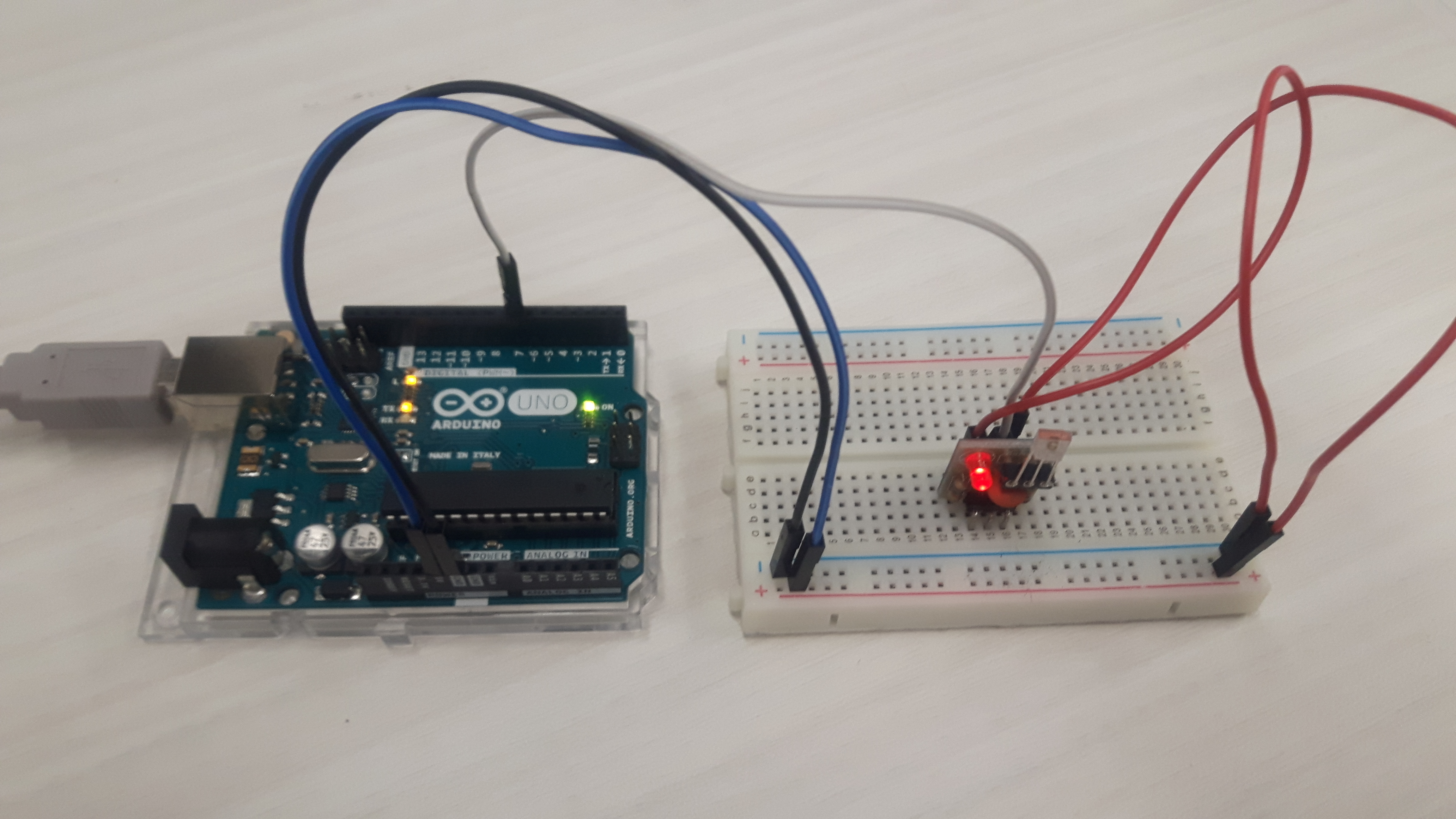
## Stumbles, Breakthroughs & Victories:

Setting up the Arduino script was one of the biggest victories that I had in this project, since I was able to reason it out and write it myself as one of the first elements I completed. After discovering Involt and using it for my project, it was a little difficult understanding which elements in their Arduino code I was allowed to manipulate and which were important, but it was a fairly easy process to replace the void setup() and void loop() functions with my own code and have it work successfully.

# Step #6 – Set Up Arduino Board

This was the second aspect of the project that I completed chronologically, after setting up the Slack webhook. With some help, I was able to connect a laser detector module that I ordered online to the Arduino board into pin 7. The set up was fairly simple, and my original code would print either 1 or 0 to the serial monitor once it was hit/not hit. As long as the board connects to power and to the correct pin, the set up is fairly simple by Arduino standards.

Below you can see a photo of the set up of the Arduino and Breadboard.



## Stumbles, Breakthroughs & Victories:

This was the simplest step of the project, as it only took less than an hour to complete, and was a great victory for me. The only stumble that I experienced was the control of the LED light that is connected to the laser detector. I was able to control the LED light or the laser detector separately, but as they both use the same pin I wasn’t able to figure out how to control the two separately. That being said, the LED wasn’t an essential aspect to my project and in the final project can’t even be seen by the user.

# Step #7 – Launching!

The final interface for the project is a bug and a target, and when you hit the target, a message is posted to Slack that says “SOMEONE FIXED A BUG!”. This console can be set up anywhere within reach of the computer, and can easily be hit by a laser pointer from across the room.

In order to run everything properly, the Arduino code has to be uploaded to the board, the Involt app needs to be open and connected to the proper port (easily done using the build-in interface), there should be a valid Internet connection, and Slack should be open (not necessary for the script to run but if you want to see the results of the laser detection it would need to be viewable). With these things, the project will be successful and you can have fun fixing bugs and telling the world about it!

