**Prototype III for the “Hot Car Emergency”**

**Deliverable H**

**Project Group B2**

Qaswar Al-Hindawi, Jacob Williams, Lauren Juneau, Navi Selvaratnam

November 28th, 2021

**1.0 Introduction**

**2.0 Prototype explanation**

**2.1 Software downloads**

2.1.1 Drivers

2.1.2 Libraries

**2.2 Software development**

2.2.1 Subsystem A: Occupant detection (HRSC-501 module)

2.2.2 Subsystem B: Temperature detection (TMP36 sensor)

2.2.3 Subsystem C: Gas detection (MQ7 sensor)

**2.3 Hardware assembly**

2.3.1 Subsystem A: Occupant detection (HRSC-501 module)

2.3.2 Subsystem B: Temperature detection (TMP36 sensor)

2.3.3 Subsystem C: Gas detection (MQ7 sensor)

2.3.3.1 MQ7 module calibration procedure

2.3.4 Subsystem D: Power supply

2.3.5 Device case design

**3.0 Prototype test plan**

3.1 Subsystem A testing and results

3.2 Subsystem B testing and results

3.3 Subsystem C testing and results

**4.0 Updated Bill of Materials**

**5.0 Feedback**

**6.0 Prototype Development**

**7.0 Limitations and Conclusion**

**8.0 Appendix 1**

# 1.0 Introduction

In the previous section, the simulated prototype sensor array circuit and codebase were modified for compatibility with the NodeMCU ESP8266 version 1.0 combined control board. The changes made to the circuit configuration and codebase were confirmed by cross-referencing available online technical documentation, and the modified prototype design was approved by the team to advance to the final stage of development. This deliverable will outline the assembly and testing of the final, physical, prototype which features the full functionality as outlined in our target specifications. The testing data compiled in this document will also include complete outlines of all relevant drivers, libraries and software used for the control board connection, as well as all the relevant calibration procedures done to ensure proper functioning of the sensor array. The final bill of materials, a description of our implementation of customer feedback and the wrike updates can be found in the last sections of this report.

# 2.0 Prototype and Explanation

## 2.1 Software Downloads

### 2.1.1 Drivers

The NodeMCU ESP8266 v1 board, as featured in our prototype, is a third party development part that is unaffiliated with Aruino or it’s partners. As such, the Arduino IDE and the Arduino Drivers are inadequate for use with the NodeMCU board. Our project utilized two drivers to achieve functionality with the ESP8266 control board:

* CH341SER USB to Serial port windows driver
* CP210x USB to UART Bridge Virtual COM Port (VCP) driver

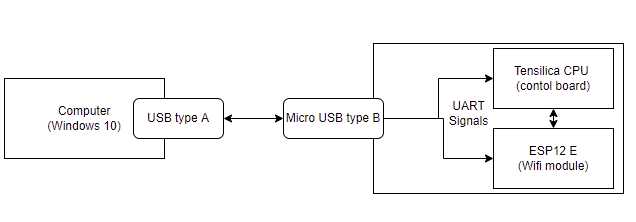
It is important to note that these drivers were downloaded onto a Windows computer using the Windows 10, version 21H2 operating system. Depending on the maker and version of it’s operating system, and it’s hardware - other computers may require different drivers to be able to interface properly with the NodeMCU ESP8266 board. The relevant information about the drivers used for this prototype are available in the table below.

| **Driver** | **Source** | **URL** | **Technical documentation** |
| --- | --- | --- | --- |
| CH341 | Nanjing Qinheng Microelectronics CO., Ltd. | <http://www.wch-ic.com/downloads/category/30.html> | <http://www.wch-ic.com/downloads/category/27.html> |
| CP210x | Silicon Laboratories, Ltd. | <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers> | <https://www.silabs.com/documents/public/application-notes/an197.pdf> |

*Table 1: Requisite driver information, download URLs and technical documentation*

#### CH341SER USB to Serial port windows driver

The CH341SER driver was downloaded to facilitate signal conversion from the input USB signal to the UART signal type used by the board for communication between it’s built-in parts. This is essential because the NodeMCU ESP8266 v1 development kit is a combined chip - it features a Tensilica 32-bit RISC CPU Xtensa LX106 as the control board and processing unit, and an ESP-12 E chip for wifi connectivity. These two chips are produced as separate components before being assembled into the ESP8266, and therefore communicate with Universal Asynchronous Receiver-Transmitter (UART) signals. There are 3 connections present when interfacing with the control board: USB type A, micro USB type B, and UART connections - the CH341SER driver allows the conversion of any one type of these signals into any other, for unimpeded communication between the board’s components, as well as between the board and the computer.

****

*Figure 1: Flow-diagram of the connections between the NodeMCU ESP board and the computer*

#### CP210x VCP Driver

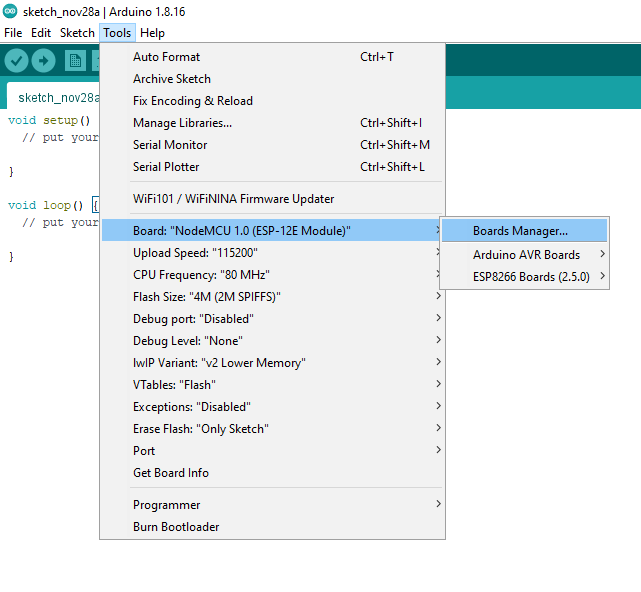
The CP210x USB to UART driver is required for device operation as a virtual COM port. When the code from the computer is uploaded to the ESP8266 chip, the computer transfers this data through a virtual COM port. The CP210x driver operates as a software add-on to the port, allowing data to be transferred to devices using software from the CP210x family.

**2.1.2 Libraries**

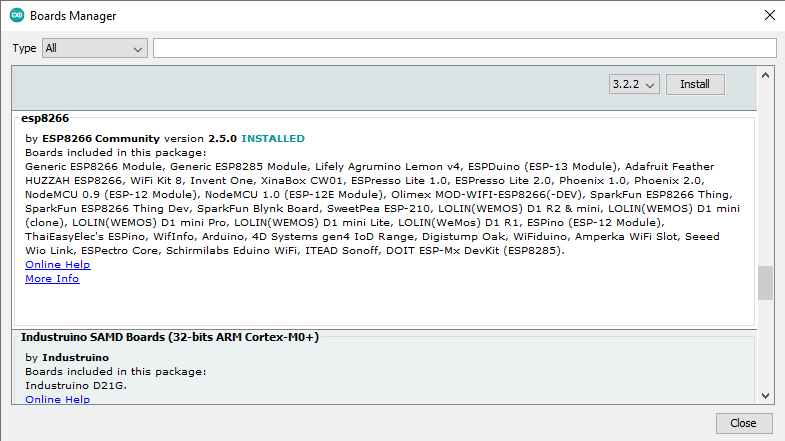
The NodeMCU ESP8266 board features a built-in ESP12 wifi module, which allows internet connectivity to the board. Because of the complex nature of wifi connectivity, a number of arduino libraries are required for adequate function. Our prototype features 3 wifi-related libraries in the codebase for the notification sending:

* ESP8226WiFi.h
* WiFiClient.h
* ESP8226WebServer.h

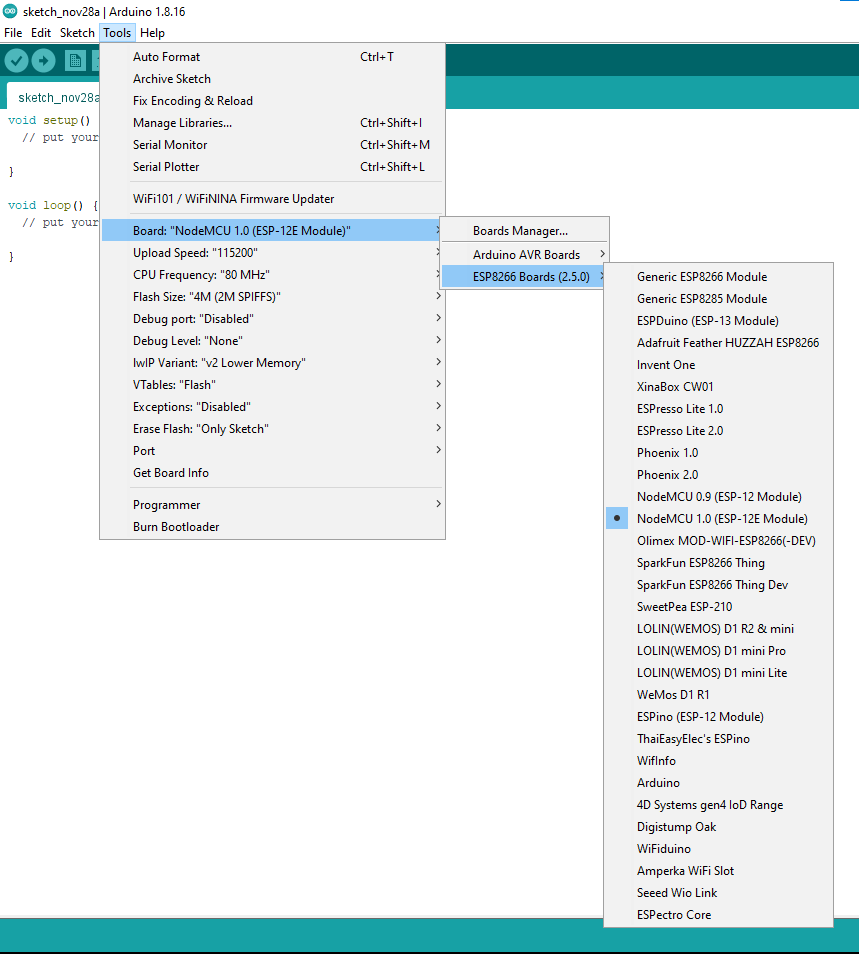
Unlike the drivers, no external downloads are required for the wifi-enabling libraries as long as the setup procedure for the ESP8226 board has been correctly executed. To setup the NodeMCU ESP8266 for use with the Arduino IDE, you must download the “esp8266 version 2.5.0” package from the board manager tab. The boards manager tab can be found on the Arduino IDE by selecting *Tools > Board: > Boards Manager…*. Screenshots of this process are available below.



*Figure 2: Board manager tab, Arduino IDE*

****

*Figure 3: ESP8266 package, in the boards manager*



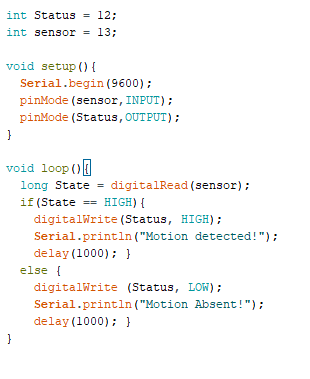
*Figure 4: Proper board selection for use of the NodeMCU ESP8266 v1*

The 3 requisite libraries will be automatically available for inclusion into any codebase once the ESP8226 package is installed. The NodeMCU board must be selected in the board manager, as described in figure 4 before use. If all of these conditions are met, the board is ready for use.

**2.0 Software development**

**2.2.1 Subsystem A: Occupant detection (HCSR-501 module)**

The HCSR-501 PIR motion sensor is an infra-red motion detection module, which outputs a digital signal. The possible return values from this sensor’s voltage output are *HIGH*, when motion is detected, and *LOW* when no motion is detected. In addition to allowing for the HCSR-501 to be connected to any available GPIO pin on the board, the digital signal output makes the code for operating this sensor extremely simple. For clarification, a snippet of code pertaining only to the PIR sensor has been provided below. The full code document for all subsystems will be available in Appendix 3.



*Figure 5: Software control for the HCSR-501 module*

The code above takes signal input from the HCSR-501, and uses it to print notifications to the serial monitor, and activate an LED when motion is detected. In the example provided here, the designation ‘12’ is used to reference pin D6, and ‘13’ is used to reference pin D7. Note also that the pinMode must still be set to INPUT/OUTPUT, as in arduino code.

**2.2.2 Subsystem B: Temperature detection (TMP36 sensor)**

The TMP36 temperature sensor outputs an analog signal which is proportional to the ambient temperature (measured in degrees Celsius). As such, it must be connected to the single analog pin on the NodeMCU ESP board. The code for this sensor is more complex than those of the other sensors, because the TMP36’s voltage output is given as a number between 0 and 1023. This single number is representative of the temperature, but must be converted before it can be expressed as degrees in celsius or fahrenheit. The conversion from raw TMP36 output values to degrees is done in several steps:

1. The raw output value is read from the analog pin using the *analogRead()* function
2. This raw value is multiplied by 3.3, the output voltage of the pin.
3. The modified value is then divided by 1024. Steps 2 and 3, combined, convert the raw output value from a simple number between 0 and 1023 into a float representing the actual voltage of the output signal in millivolts.
4. From there, 0.5 is subtracted from the value to account for the sensor’s offset. This offset allows the temperature sensor to express negative degree temperatures with it’s strictly-positive 0 to 1023 raw output.
5. Finally, the value is multiplied by 100 to get degrees celsius. Optionally, a conversion to fahrenheit can be inserted here, as was done in our example.

For clarification, a snippet of code pertaining only to the TMP36 sensor has been provided below. The full code document for all subsystems will be available in Appendix 3.

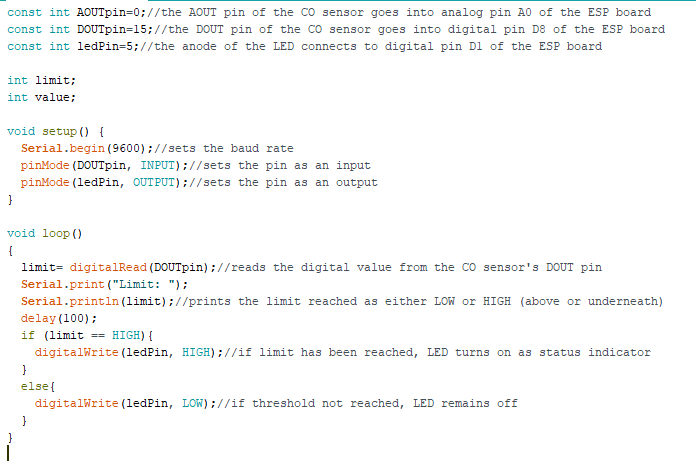


*Figure 6: TMP36 sensor code*

In our example, an extra division has been added to the final step of the conversion to celsius, as our TMP36 is installed in a sensor brick whose PCB modifies the voltage output by a small margin. The conversion from raw sensor values to degrees is by far the most complex part of this sensor’s code. The sensor’s values are easily read by the *analogRead()* function and the pin designation must be “A0” for the only analog pin. Note as well that the baud rate of this code is set to 115200. This sample speed allows for more signals to be sent and interpreted for any given timeframe, allowing the sensor greater reactivity to changes in temperature.

**2.2.3 Subsystem C: Gas detection (MQ7 sensor)**

The MQ7 gas sensor features pins allowing for both Digital and Analog output signals. The analog output returns a voltage proportional to the concentration of ambient carbon monoxide, and the digital output changes from *LOW* (rest value) to *HIGH* (activation value) based on an internally configurable threshold value. Depending on the configuration of the sensor, both of these values may be utilized at the same time. Unfortunately, the NodeMCU ESP board has only one analog pin, which has been assigned to the TMP36 temperature sensor. A multiplexer unit would therefore be required to make use of the MQ7’s analog output, which was not allowed for in our budget or timeframe. However, the digital output of the gas sensor can be utilized for our purposes as long as the digital threshold is adequately set with a resistor in the circuit. Section 2.3.3 has more information about the physical wiring of the MQ7. For clarification, a snippet of code pertaining only to the MQ7 sensor has been provided below. The full code document for all subsystems will be available in Appendix 3.



*Figure 7: MQ7 code*

**2.2.4 Subsystem D: Mobile connectivity (Esp8266 wifi module & Pushingbox/Pushbullet)**

The goal behind this subsystem is to notify the users of our device ( by sending push-notifications on their mobile phone) if there is any repeated movement in their car while they are not around. Since we use the NodeMCU ESP8266 v1 board as our control board, the board comes with a built-in ESP wifi module that can be programmed to connect one's wifi connection using their SSID and password. Using that wifi connection, our device will send the information regarding temperature, movement, and CO directly to an online cloud service called Pushingbox, which can send notifications based on API calls.

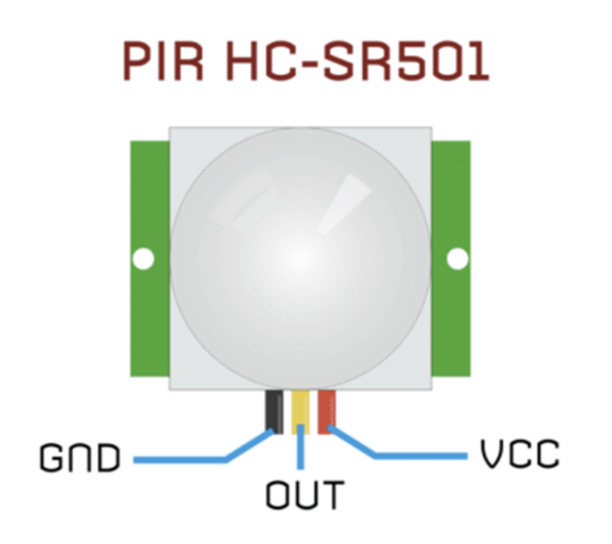
The Pushingbox services can connect directly to a mobile application called Pushbullet, which the user will have to download onto their phone. We have made two different scenarios that will send two different mobile notifications if the requirements are met.

1. If the PIR sensor detects repeated motion, but the conditions are safe.
2. If the PIR sensor detects repeated motion, but the conditions are dangerous.

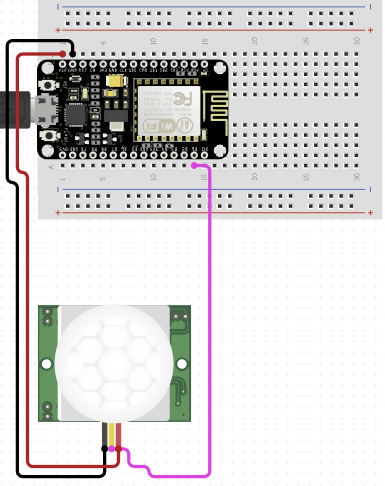
By having these two scenarios, we make sure that the user of our device knows if there is movement inside his car and knows if the conditions inside the vehicle are safe. The code that implements these actions will be posted in the appendix.

**2.3.1 Subsystem A assembly (PIR motion sensor)**

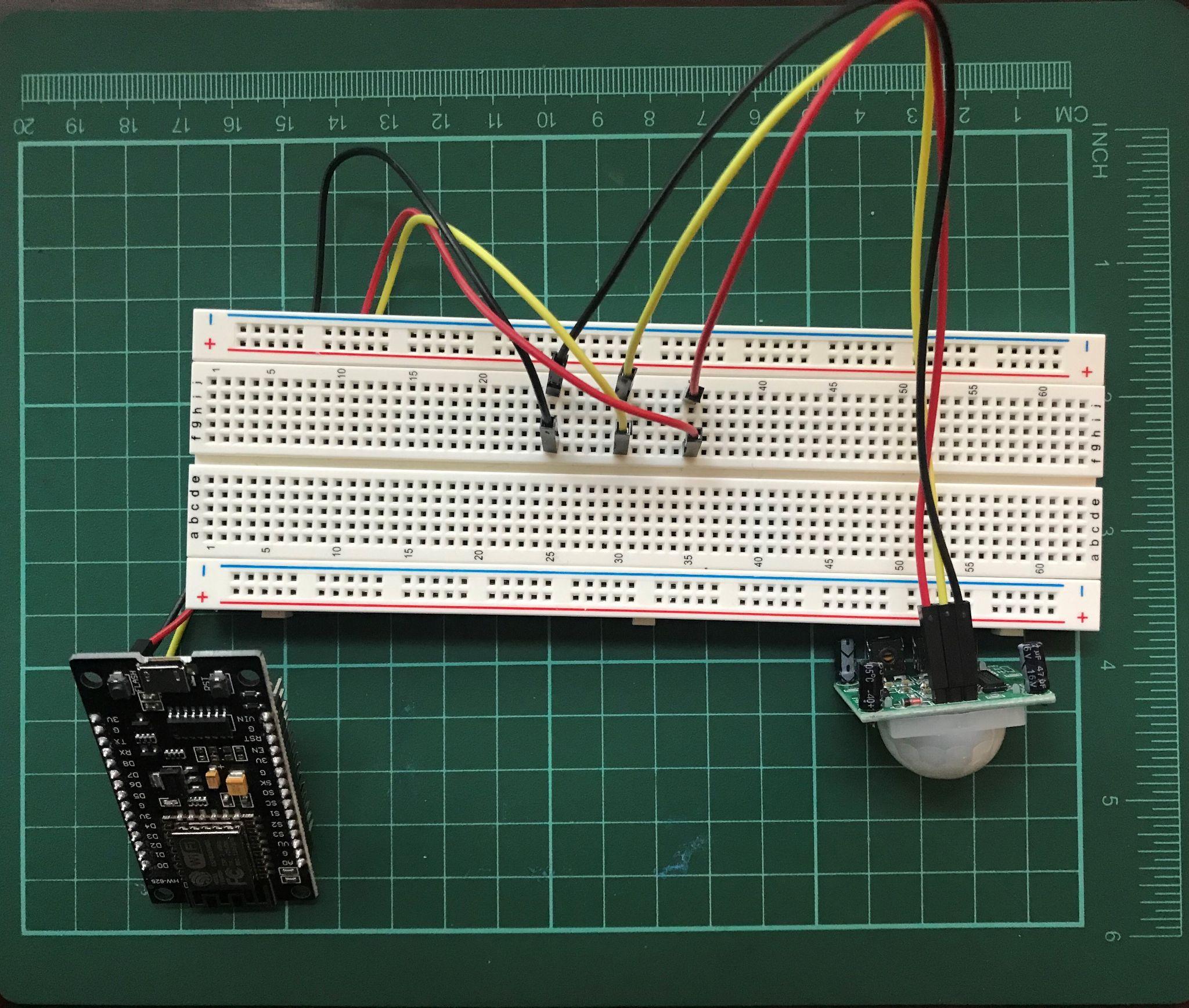
The PIR motion sensor used in our final prototype is the HCSR-501 module. This sensor, as used in our prototype, is a standalone module with no accompanying accessories. It features 3 connections: (from left to right) a Ground pin, the Digital Voltage output pin, and the 3.3V Voltage input pin. During the prototyping phase, these pins were connected to the breadboard via female-to-male connection wires. The output, in our case, was connected to the D7 pin on the NodeMCU ESP8266 board. For clarity in the image, the PIR sensor’s circuit is shown alone on the breadboard, a photo of all sensor subsystems wired together will be available in Appendix 2.



*Figure 8: HCSR-501 PIR motion sensor pin layout*

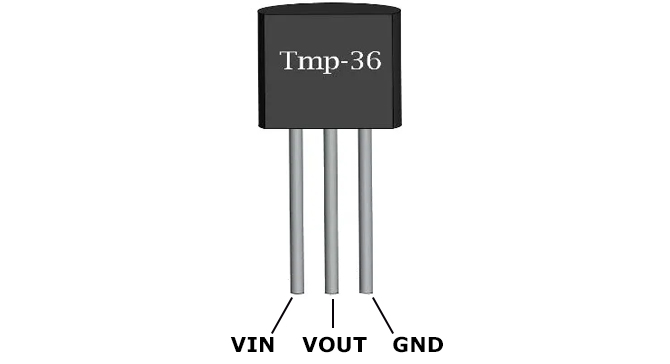
****

*Figure 9: HCSR-501 PIR motion sensor circuit diagram*

**

*Figure 10: HCSR-501 PIR sensor circuit*

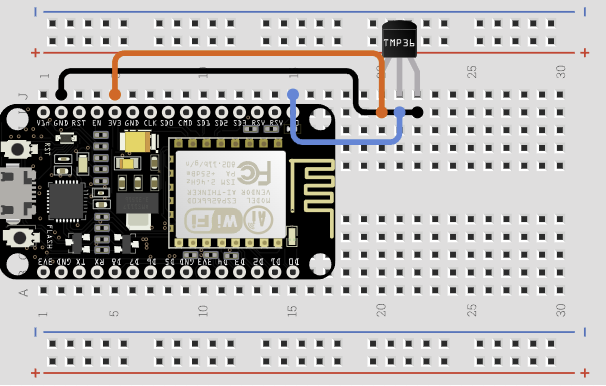
**2.3.2 Subsystem B assembly (TMP36 temperature sensor)**

Our prototype features it’s temperature sensor in the form of the Octopus TMP36 temperature sensor brick. Within the configuration of the octopus sensor brick, the TMP36 is mounted on a PCB-base and connected via male-to-female pin-terminal wires to the output. During the prototyping phase, external male-to-male wires were required to connect the sensor brick to a breadboard. 

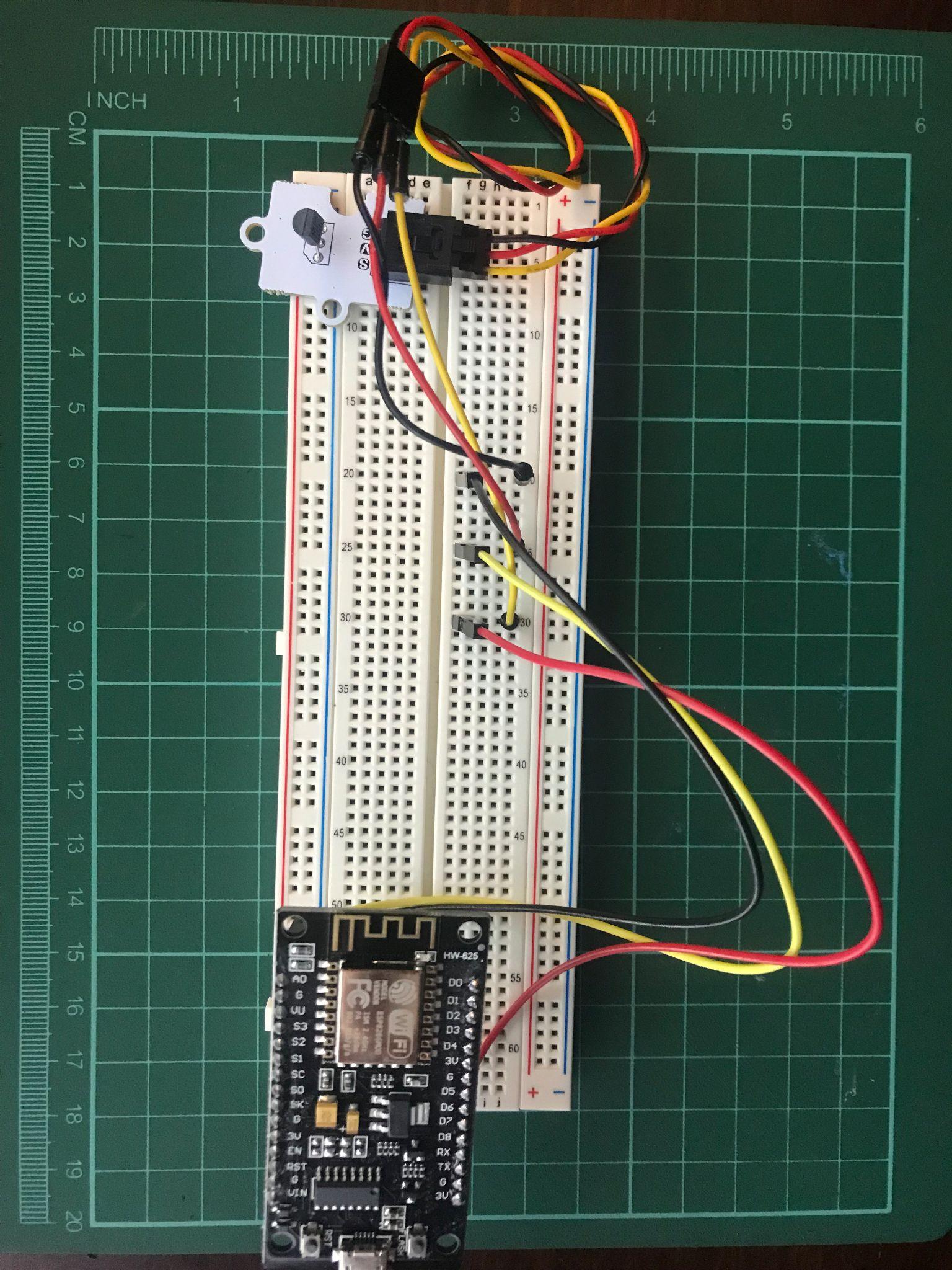
****

*Figure 11: The Octopus Temperature sensor brick, and TMP36 pin layout*

The TMP36 sensor brick has 3 connections: Ground (denoted by ‘G’ on the PCB), Power in (denoted by ‘S’ on the PCB) and Voltage output (denoted by ‘V’ on the PCB). Since the TMP36 temperature sensor outputs an analog signal output, the ‘V’ pin was connected to the A0 pin on the NodeMCU ESP8266, as seen in both the circuit diagram and the photograph below. For clarity, the photo of the physical circuitry of the TMP36 sensor array is shown on it’s own - a photo of all subsystems wired together will be available in Appendix 2.

****

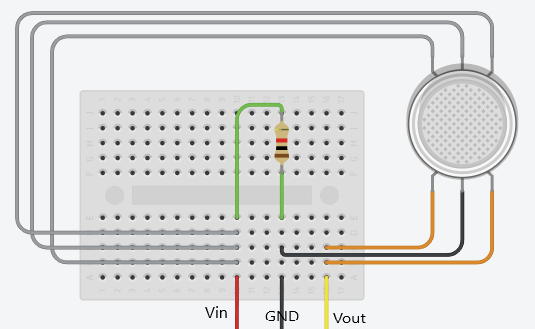
*Figure 12: TMP36 sensor circuit diagram*

****

*Figure 13: TMP36 Physical circuit (with breadboard)*

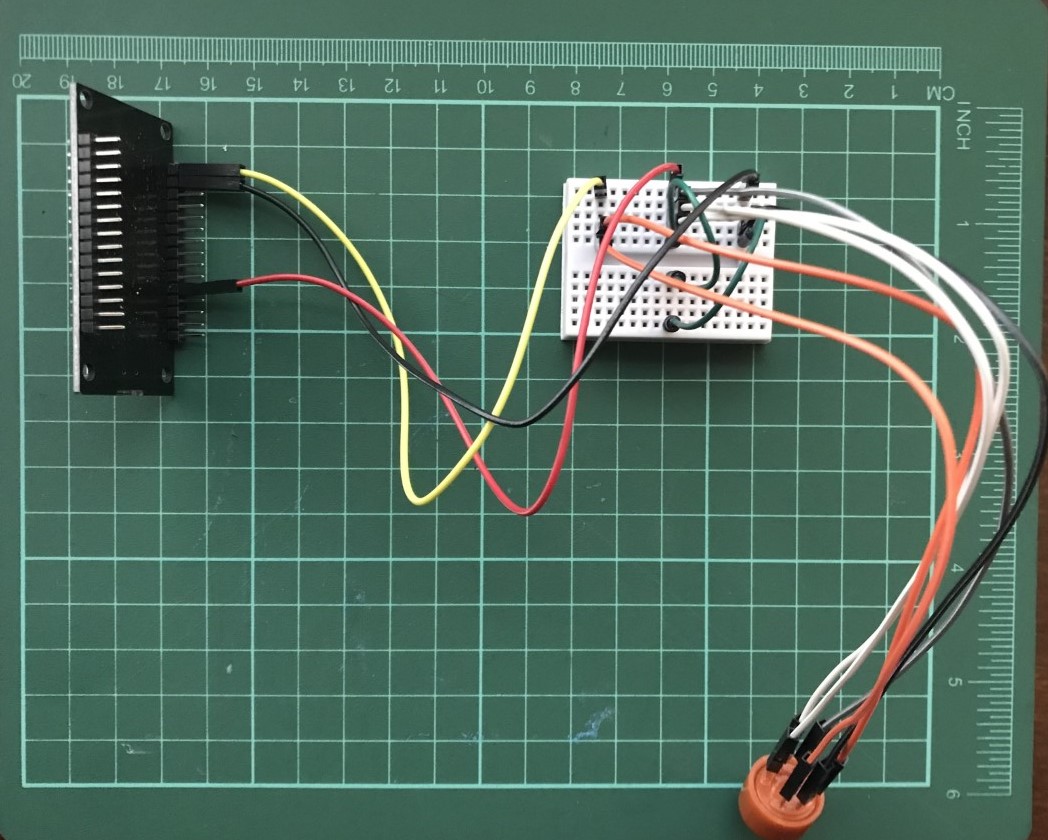
**2.3.3 Subsystem C assembly (MQ7 gas sensor)**

The gas sensor component of the sensor array has, by far, the most complex wiring in the entire prototype. This is due mainly to the fact that the bare MQ7 sensor has 6 output pins. Commercial PCB breakout boards are available to reduce the MQ7’s 6 pin output to 3 or 4 pin outputs, but none were available to this project. To simplify the circuit, then, a breakout board analogue was fashioned with a breadboard and connection wires.

****

*Figure 14: Circuit diagram for the MQ7 sensor’s secondary wiring*

By utilizing the circuitry above, the 6-pin output for the MQ7 sensor is simplified to 3 output pins: Ground (Black), 5V power input (Red), and Voltage output (Yellow). The resistor shown on the board regulates the sensitivity of the sensor. Our configuration features a 10kΩ threshold sensitivity-setting resistor. With 10kΩ resistance, the built-in threshold is set to 100 parts-per-million (PPM). This secondary wiring is then connected to the NodeMCU ESP8266 control board, with the voltage output connected to the A0 analogue pin. To simplify this already very complex circuit, it is shown on it’s own - a photo of all subsystems wired together will be available in Appendix 2.



*Figure 15: Gas sensor circuit*

**Our prototype’s limitation: The MQ7 gas sensor**

An important limitation in our design arises when one inspects the MQ7’s technical data. Internally, the MQ7 gas sensor is a semiconductor whose resistance changes according to ambient concentrations of Carbon monoxide molecules. Measurements are taken by measuring differences in resistance of the semiconductor as it changes in the presence of CO molecules. The semiconductor in the gas sensor operates at high temperatures, and as such features an on-board heater to maintain the requisite temperature in the sensor. The MQ7’s heater operates on cycles: a low-voltage cycle is utilized to warm the sensor for data collection, and a high-voltage cycle is utilized to purge the sensor of CO gas to ensure accurate measurements on the next low-voltage cycle. The low-voltage cycle requires 1.4v, which is perfectly within the 3.3 volt pin output for the NodeMCU ESP8266 board. The high-voltage cycle, however, requires 5 volts - which is well beyond the capabilities of the NodeMCU’s input/output pins. There are several possible solutions to this problem, some of which we will discuss below.

| **Possible Solution** | **Explanation** | **Price & Attainability** |
| --- | --- | --- |
| Use of a Voltage Divider | To attain the necessary voltage to safely use the MQ7, we can use a voltage divider. One can supply the sensor with the necessary voltage, then use the voltage divider before the connection to the NodeMCU ESP8266. | This is an example of a 5V voltage divider you cam purchase at pishop (where our other parts came from). However, it will run you about $10.00  <https://www.pishop.ca/product/pololu-5v-2-5a-step-down-voltage-regulator-d24v22f5/> |
| External ADC | Use an analogue to digital converter that can handle the necessary voltage. The example on the right can handle 2.7V to 5.5V which is perfect. Connect the ADC to the microcontroller, then analogue values of 2.7V to 5.5V can be read. | This is a good option of ADC found at digikey. Just under $2.00, which would still put us over budget.  <https://www.digikey.ca/en/products/detail/microchip-technology/MCP3021A5T-E-OT/593685> |

*Table 2: Solutions to the MQ7 Limitation*

**2.3.4: Power supply**

The NodeMCU ESP8266 board features 3 possible power inputs: the micro-USB connection, the Vin I/O pin, and the 3V I/O pin. The micro-USB and the Vin connections are linked to the board’s built-in voltage regulator - an AMS1117 1 amp low dropout voltage regulator. The control unit on the NodeMCU board operates on a 3.3 volt power supply, which is supplied by the voltage regulator while operating under power from the USB connection. In order to produce adequate amperage for the function of the board, any power supply connected to the Vin pin must be at least 7 volts, and ideally must not exceed 12 volts - the theoretical maximum voltage allowed for by the voltage regulator is 15 volts, but operating at this maximum voltage for extended periods may negatively impact the lifespan of the AMS1117. The final power input, the 3V pin, is unregulated. This means that power inputs to the 3V pin must be between 2.3 and 3.6 volts, with a typical value of around 3.3.

**AA Batteries:**

AA batteries were initially chosen as a power source for our prototype, but there are certain limitations that present themselves with this type of battery. Primarily, Alkaline AA batteries have a lower energy density than other possible batteries with comparable discharge voltages. A number of different configurations of AA batteries is possible as a power source for the NodeMCU ESP board, and in this section we will discuss a few possibilities.

A combination of two AA batteries in series have a working voltage of 3V, which is adequate for powering the NodeMCU board from the 3V pin. However, at 50% capacity, the discharge voltage of the batteries decreases to around 2.4 volts, which is unacceptably close to the 3V pin’s minimum value of 2.3 volts. Additionally, the NodeMCU board draws 250 mA of power, which would drain the AA battery pack in around 8 to 12 hours (for a standard 2000-3000 mAh battery) without the use of any sensors. This configuration was ultimately utilized for our prototype, because of its use of our already existing battery holder, it’s compact size, it’s simplicity, and it’s capacity (which is adequate for a demonstration).

A combination of four AA batteries in series (as is found with the largest AA battery holder available from the makerspace store) have a working voltage of 6V, which is sufficient to power the board, but is unsuitable for both the 3V pin and the Vin pin connections. A 6-volt power supply connected to the 3V pin would damage onboard components, and 6 volts are insufficient to generate adequate amperage when connected to the Vin pin. For this configuration to work, the voltage must be either dropped to 3 volts or increased to at least 7 volts. Dropping the voltage can be achieved using resistors or by fitting a voltage regulator.

A combination of six to eight AA batteries would have working voltages between 9 and 12 volts, which are in range for the Vin pin’s AMX1117 voltage regulator.

**E / PP3 9V Batteries:**

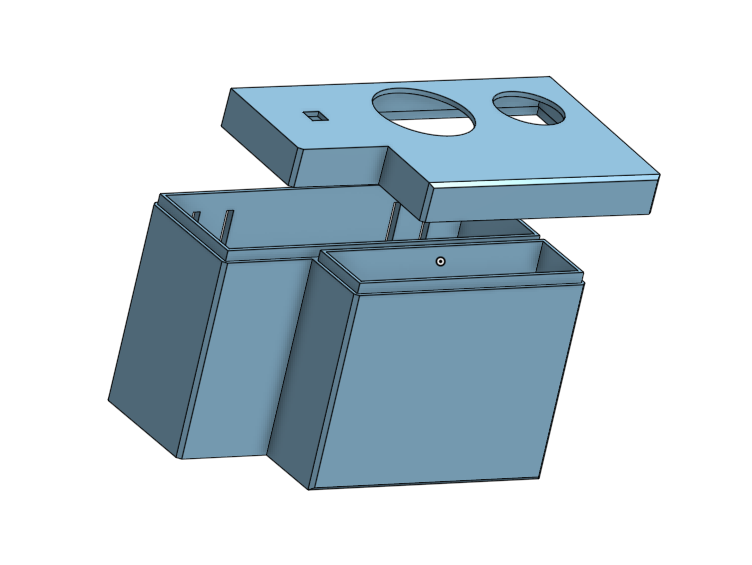
The simplest solution for operating the NodeMCU board from an alkaline battery power source is the use of a 9-volt rectangular format battery. These batteries are common and provide adequate voltage without requiring any additional resistors or voltage regulators. It is extremely important to note, however, that a standard alkaline 9V E/PP3 battery has a capacity of only around 500 mAh. This is a disappointing figure, as it means that the NodeMCU board will completely discharge the battery on it’s own in little more than 2 hours. For any purposes other than a brief technical demonstration, this battery would be inadequate to power our device.

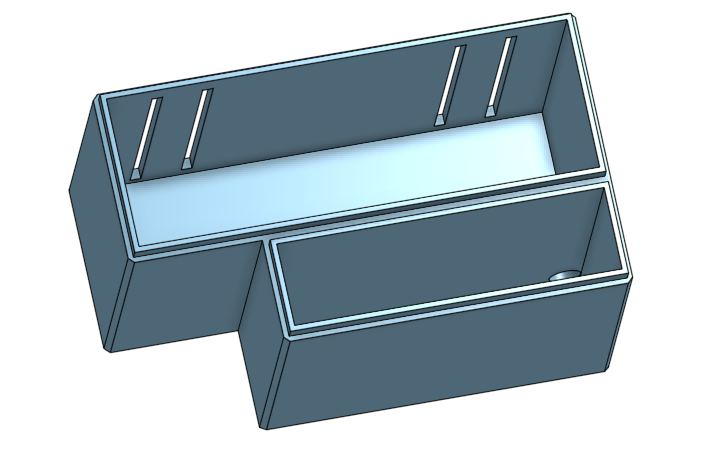
**Lithium-polymer and Lithium-ion batteries:**

For the long-term operation of a device such as ours, the natural choice of power supply is rechargeable batteries. Lithium-polymer and lithium-ion batteries can vary somewhat, but often have discharging voltages between 2.7 volts and 4.2 volts. Of the lithium-core batteries considered for this project, none were able to be connected directly to the 3V pin and none produced adequate voltage to be connected to the Vin pins. This means that additional hardware is required for any configuration featuring these types of batteries as a power source for the device. The additional electronics required for this solution were outside of our budget, which is why they weren’t utilized for our prototype.

**2.3.5: Device Case Design**

As a result of continuous change throughout the iterative design process, the original 3D render on OnShape in the previous deliverables is insufficient for the current prototype. Below is the final case concept designed using OnShape.





*Figure 16: Final render of device case. Bottom left is the lid, bottom right houses the circuit and batteries*

This render was developed with the dimensions of our sensors, battery pack, and the NodeMCU ESP8266 in mind. However, since this group only has one member in Ottawa available to print the case, where one member who lives further out is working on the circuitry, the decision was made to wait until the circuitry was done to print the case to save time and material.

# 3.0 Prototyping Test Plan

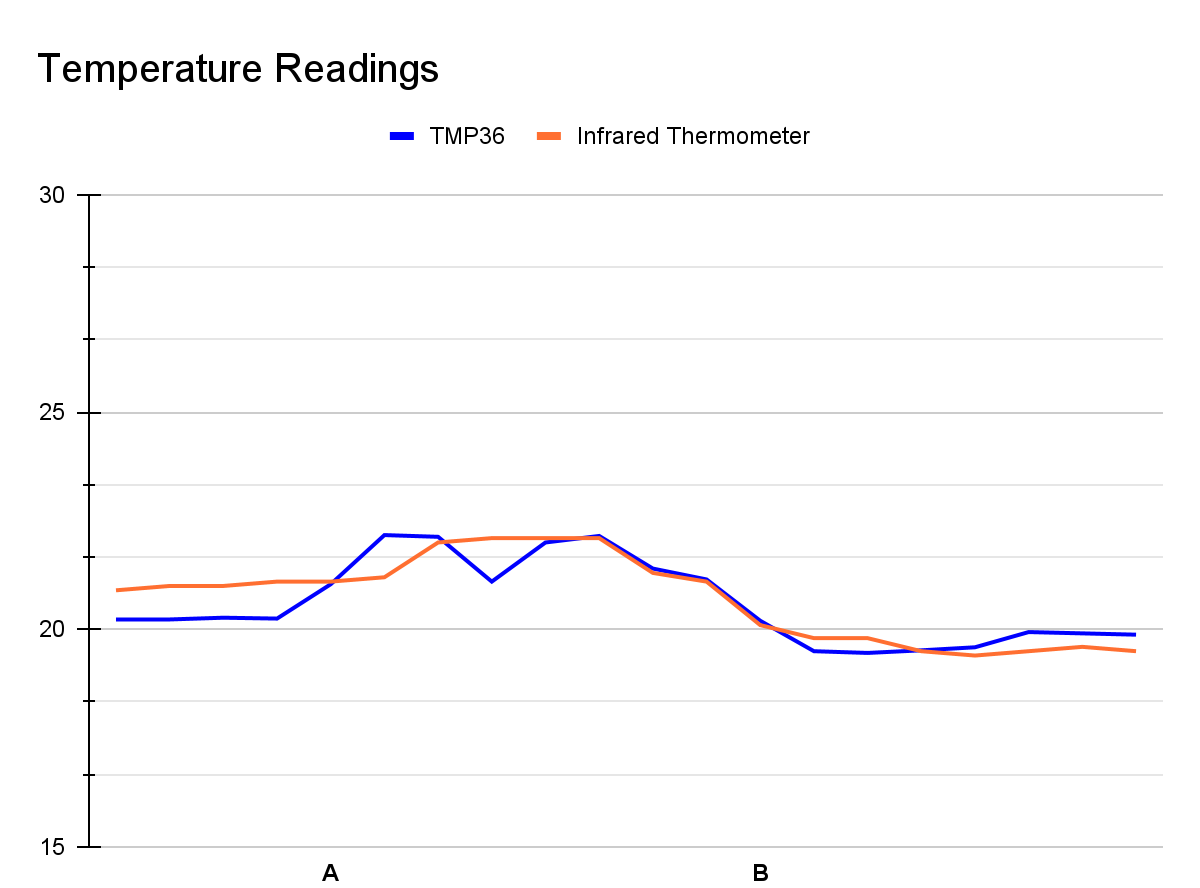
The final phase of prototyping encompasses all of the subsystems and circuitry. After this deliverable, the only remaining concern is joining the circuitry with the case as our group members do not live near each other and cannot often exchange parts. Test 3.2 and 3.8 were removed. , and 3.8 was removed as the test on physical concept was more than adequate from the last deliverable. 3.5 and 3.6 weren’t possible due to our limitations with the MQ7 probe.

| **Test** | **Part & Subsystem** | **Objective** | **Test Method** | **How Results will be Recorded** | **Date** |
| --- | --- | --- | --- | --- | --- |
| **~~Prototype I: Analytical Focused on Circuitry Virtual Prototype~~** | | | | | |
| **~~Prototype II: More Complex Virtual Circuitry and Coding, Physical Concept~~** | | | | | |
| **Prototype III: Working Circuitry and Physical Concept** | | | | | |
| **3.1** | Pushbullet device connection  Mobile connectivity | A key portion of this project is notifying a parent/guardian/car owner that a child is present in the vehicle during dangerous conditions. This test will isolate the feasibility of doing so. | Sample information and notifications will be sent to team member’s phones | Screenshots of the shared sample information and notifications will be recorded from the phones | Nov 19 - Nov 22 |
| **3.3** | PIR Motion Sensor (HCSR-501) Code and Circuit Test (Physical Circuitry)  Occupant detection | To verify the circuit and sensor assembly to ensure adequate function | Objects will be placed in front of the device at ½ meter increments until it’s operating range. These objects will be moved at varying speeds at different times to test the sensor’s speed and accuracy | Sensor response times, object distances and velocities, and sensor outputs will be recorded in a spreadsheet | Nov 17 - Nov 19 |
| **3.4** | Thermistor Sensor (NTC) Code and Circuit Test (Physical Circuitry)  Temperature detection | To verify the circuit and sensor assembly to ensure adequate function | The sensor will be placed in an environment with a controlled temperature. The temperature will then be incrementally increased from 0 degrees celsius until the upper bound of the thermistor’s operating range. | Sensor outputs will be recorded in a spreadsheet and compared with control values. | Nov 17 - Nov 19 |
| **3.7** | LED Array or Buzzer System Code and Circuit Test (Physical Circuitry)  Alarm | To verify the circuit and sensor assembly to ensure adequate function | Buzzers will be placed at incremental distances from a microphone to determine their relative power | The volume of each buzzer will be measured in decibels and compared with each other to establish relative usefulness and volume | Nov 19 - Nov 22 |

*Table 3: Updated Prototyping Test Plan*

**3.1 Subsystem A testing and results**

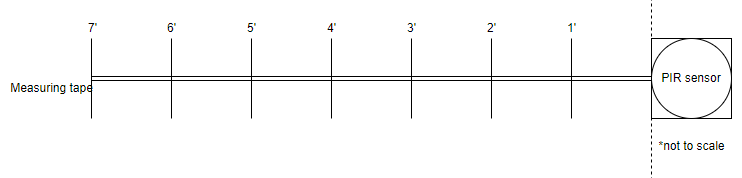
The temperature sensor was tested in a controlled environment. Readings were recorded in a dry, insulated room, and compared with data collected with an external thermometer. The verification thermometer was an NX2000 Infrared Thermometer, it’s technical documentation is available in Appendix 1. The NX2000 thermometer was configured to measure the temperature of objects within its range, and it’s data was compiled by using it to measure the ‘object temperature’ of the TMP36 sensor. After beginning the experiment and allowing the readings to normalize, the temperature was changed by covering the sensor with a finger to measure it’s sensibility to slight temperature changes. The graph below shows the sensor’s readings compared to those collected from the same surface with the infrared thermometer. Point A denotes when the temperature manipulation began, and Point B denotes when it ended.



*Figure 17: TMP36 readings vs. IR thermometer readings*

**3.2 Subsystem B testing and results**

The PIR motion sensor was tested in a controlled environment. Movement readings of a person were collected for a variety of distances and angles from the sensor, to test it’s range and field of view. Distances were measured with a measuring tape placed on the floor of a long room, with the PIR sensor at the beginning of the graduations. Readings were taken in 1-foot increments from 1 foot in front of the sensor out to 7 feet (the theoretical maximal distance of the sensor). A recreation of the testing conditions are available under figure 18.

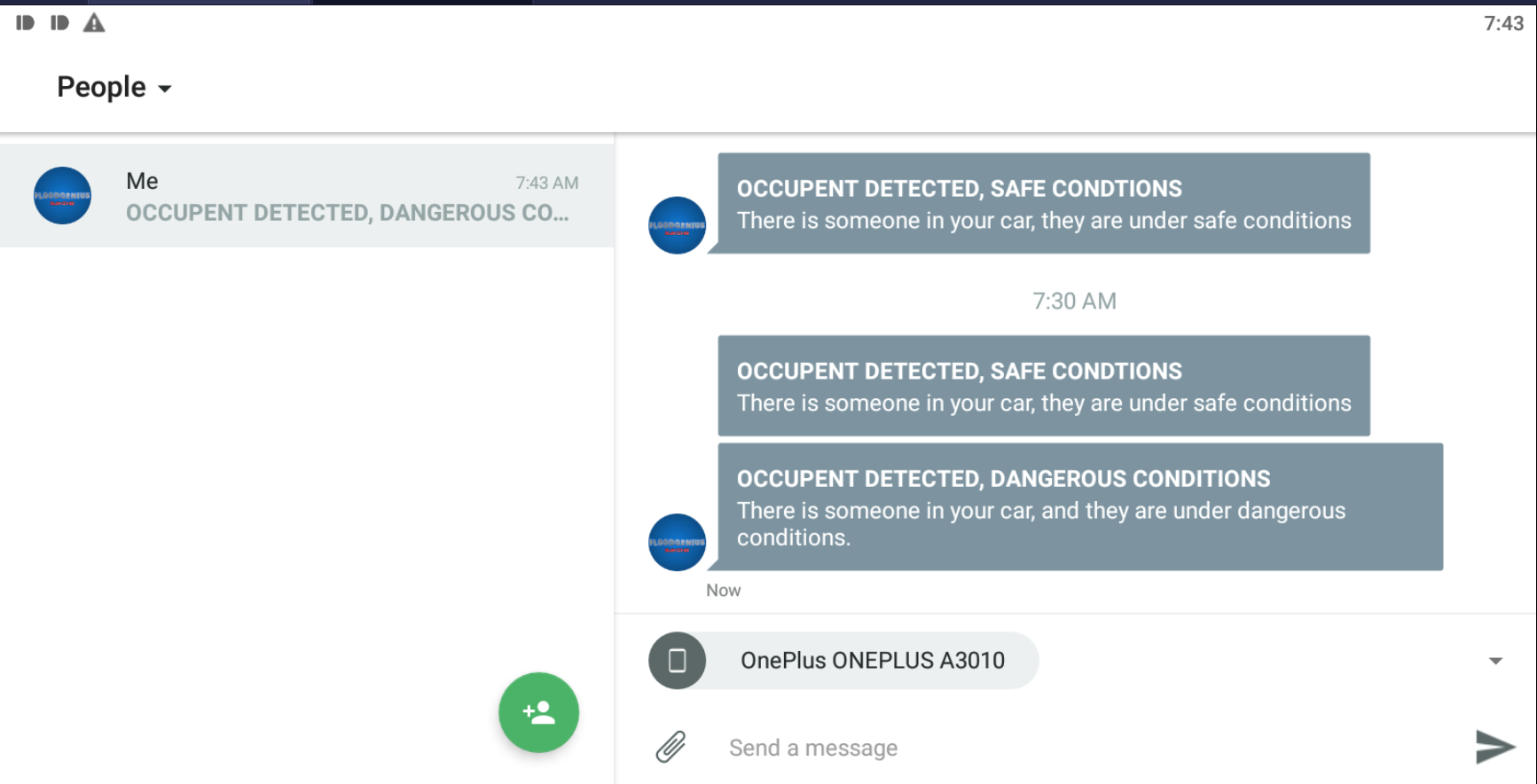


*Figure 18: Recreation of the PIR motion sensor’s testing configuration*

Testing showed that the PIR motion sensor was able to detect motion at a distance of 4 feet

**3.3 Mobile connectivity testing**

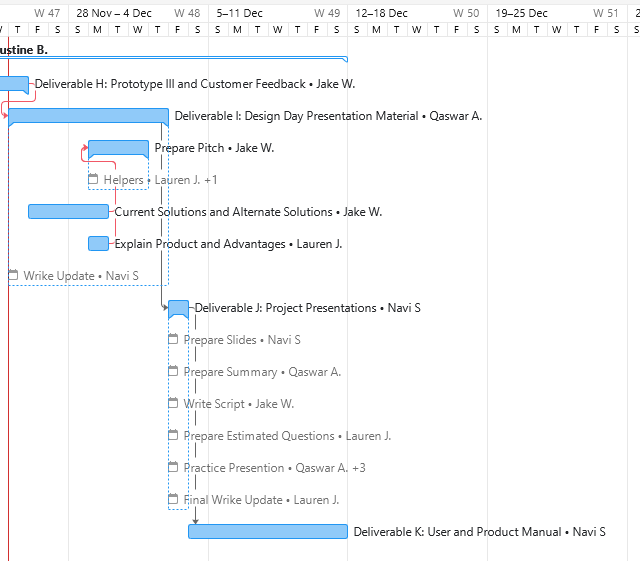
The mobile connectivity testing was conducted with the fully assembled board and circuit. The figures below are screenshots of the Pushbullet service receiving notifications from our prototype.



*Figure 19: Pushbullet notifications*

**3.2 Wrike Update**

Below is the weekly wrike update for design day and the project presentation. Since there is no formal due date presented on the BrightSpace for the final presentation, it was impossible to set proper dates on the wrike. Therefore, the estimated length of time for each task follows. Overall deliverable J is estimated to take 4-5 days, once a due date has been given out, we will start working on the corresponding date.



*Figure 20: Updated Wrike Gantt Chart*

| **Task** | **Estimated Length** |
| --- | --- |
| Prepare Slides | 3-4 days |
| Prepare Summary | 2-3 days |
| Write Script | 3-4 days |
| Prepare Estimated Questions | 1 day |
| Practice Presentation | 1-2 days |
| Final Wrike Update | 1 day |

*Table 4: Estimated length of time for deliverable J tasks*

# 4.0 Updated Bill of Materials

Below is the final update of the bill of materials. Changes made are highlighted in yellow. Essentially, two items had been omitted from the original version as an error as at the time, the physical design of the case was less of a priority.

| **Part** | **Description** | **Price** | **Store** | **Url** |
| --- | --- | --- | --- | --- |
| HSCR-501 | PIR motion detecting module | $3.95 | Pishop | <https://www.pishop.ca/product/hc-sr501-pyroelectric-infrared-pir-motion-sensor-detector-module/> |
| TMP36 | Temperature sensor | $3.45 | Pishop | <https://www.pishop.ca/product/octopus-temperature-sensor-brick-tmp36-analog-for-microbit/> |
| MQ7 module | CO gas sensing module | $9.95 | Pishop | <https://www.pishop.ca/product/carbon-monoxide-gas-sensor-mq-7/> |
| AA Batteries | Standard AA batteries for power supply | $2.85 | Pishop | <https://www.pishop.ca/product/aa-battery/> |
| Battery holder | Case to house the AA batteries | $1.00 | Makerspace | <https://edu-makerlab2021.odoo.com/shop/product/aa-battery-holder-48#attr=47> |
| ESP8266 ESP-12E v1.0 | Combined control board and wifi module to control modules and send push notifications | $9.95 | Pishop | <https://www.pishop.ca/product/esp8266-esp-12e-v1-0-wifi-cp2102-iot-lua-267-for-nodemcu/> |
| Protoboard | Board to solder circuits | $0.50 | Makerspace | <https://edu-makerlab2021.odoo.com/shop/product/protoboard-51?search=board#attr=53> |
| Resistors  (220Ω,4.7kΩ) | Resistors for the LEDs and the MQ7 module | $0.02  (1¢ \*2) | Makerspace | <https://edu-makerlab2021.odoo.com/shop/product/resistor-6?search=resistor#attr=9> |
| Connection wires | Wires for connecting the various components | $2.50 | Makerspace | <https://edu-makerlab2021.odoo.com/shop/product/wire-5ft-45?search=wire#attr=213,217> |
| Elastic straps | Elastic straps that will connect to the device in order for it to fasten onto the arm rest of the car. | <$2.00 | Dollarama | n/a |
| ABS Plastic (3D printing) | Plastic used to 3D print the casing | <$1.00 | Makerspace | n/a |
| **Total:** | **$41.55 After tax, $49.05 After shipping** | | | |

*Table 4: Update BOM*

# 5.0 Feedback

Feedback is valued very highly in the iterative process. Throughout the course of the past 3 deliverables, we have acquired a lot of feedback. Some has already been discussed in previous deliverables as beyond the scope of our abilities, budget, and time. For valid concerns, such as ‘how the product would respond to varying temperatures’ was analyzed during production. It is for this reason, for example, the device does not sit on the dash exposed to direct sunlight. There is nothing more important than obtaining potential user/customer feedback. We implemented the previously received feedback into our updated product such being, the stability and sturdiness of the overall product and for that we have of course implemented the straps to secure the location of the product. As well as with the fact that the overall case of the product will be 3D printed using ABS plastic. Of course with a higher budget the materials can be adapted and improved. In regards to concerns as ‘water damage’ it is highly recommended that water not get on/within the product. However as seen by figure 16 , the design of the case, it is built in a way that secures the wires in place, safely and thus should be able to withstand a minimal resistance to water.

## **5.0.1 Individual** 2 (again) **:** Engineering student at Carleton, licensed driver, with a younger sibling

* *I’m concerned that the mechanism for attaching it to the vehicle is a problem. I don’t think opening armrests are standard in every vehicle. I guess you could use the elastic straps to attach it to somewhere else.*
* *Will an LED actually be good at notifying a bystander regarding a potential problem? I mean, it’s good for testing purposes..*
* *It might be nice to have a dead battery indicator or an on and off switch.*

## **5.0.**2 **Individual 7: Part-time worker, licensed driver**

* *I worry about how resistant the product is to varying temperatures. As well as how resistant the device is to water damage.*

# 6.0 Prototype Development

This is our third and final prototype, and it has come a long way since the first. In the first round of testing, we created and perfected our code using TinkerCAD to mitigate issues once we received our parts. In the second round of testing, we found the most optimal spot for the device in the car based on a colloquial analysis of its effect on sensors and in general, how much ‘in the way’ it is. We also verified the code for the new language we needed to use for NodeMCU ESP8266. And now, in this deliverable, we used the established code and connected the sensors to the ESP8266 and developed a new case based on the testing data from the second prototype. We rigorously tested our motion and temperature sensors, as well as set up a secure push notification mechanism and ensured its functionality. This prototype was the textbook definition of a successful iterative process.

# 7.0 Limitations and Conclusion

Admittedly, our third prototype is not as comprehensive as we would have liked. We suffered many challenges and limitations as a result of part order issues, and notably, proximity to group members. It is challenging for one person to strong hand the circuitry without hiccups. Nevertheless, in this deliverable we have developed as comprehensible a prototype as possible. A device case was rendered to meet the needs of the functional circuitry. Each subsystem was thoroughly tested as outlined in our prototype testing plan, and the circuitry was finalized for insertion into the case. We reached a heavy limitation with regards to our CO sensor. This issue was outlined in depth in section 2.3.3. Unfortunately due to time restrictions, we have to move forward with the MQ7 sensor not being functional. In this deliverable, we outline alternatives to remedy this issue, but they are not within the scope of our time and budget. Furthermore, we have updated our bill or materials and prototype test plan, as well as reached out to our community for feedback on our prototype to optimize the iterative process. With our limitations in mind, the group has minimal tasks before design day; combination of case and circuitry, and finalizing the presentation.

**8.0 Appendix 1**

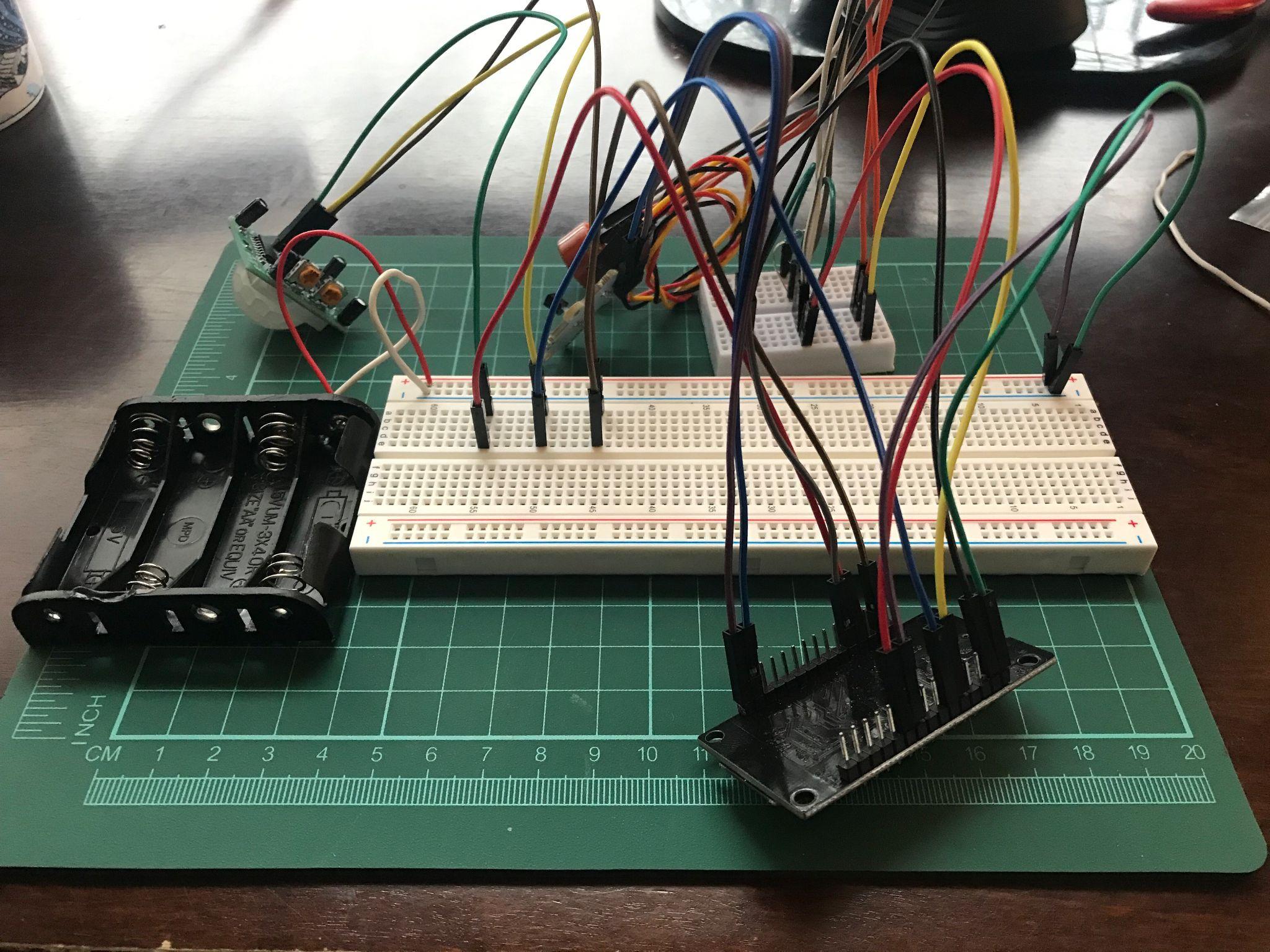
**The Clipper corporation NX2000 Infrared Thermometer**

This thermometer is AAA-powered, and is able to measure temperature in Celcius and Farenheit between 00C and 400C with an accuracy of ±0.30C (when in “object TEMP” mode). It is operated by using specialized lenses in front of the sensors that direct infra-red light into a thermopile, which absorbs the infrared light converting it into heat. A voltage output proportional to the temperature is outputted, and used to determine temperature readings. It is worth noting that this thermometer is designed for use in detecting body temperature, and as such it’s uncertainty is minimized within the 350C to 420C range.

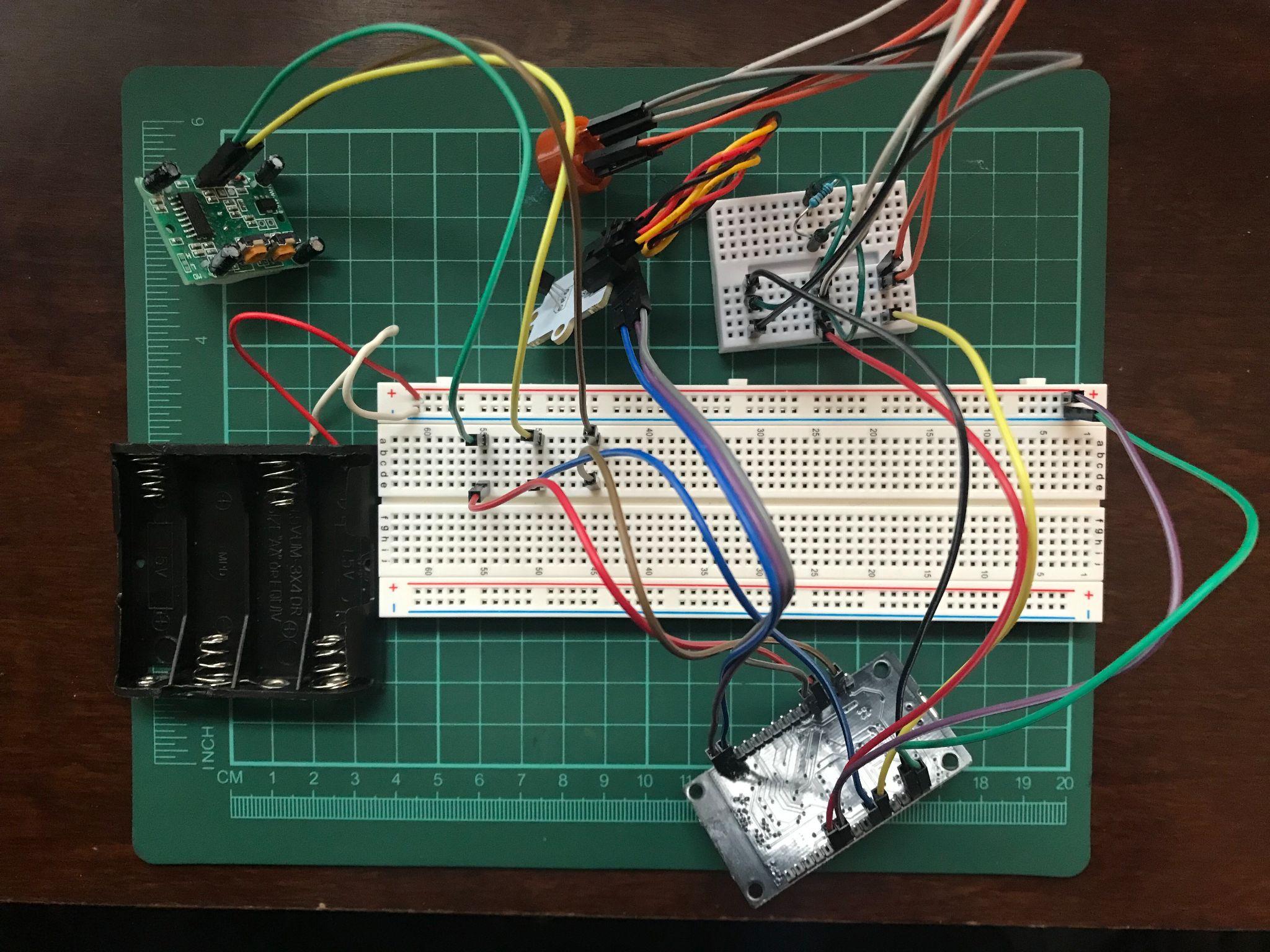


*Figure 21: The Clipper NX2000,* [*https://www.wasserstrom.com/restaurant-supplies-equipment/thermometer-forehead-infrared-noncontact-6085215*](https://www.wasserstrom.com/restaurant-supplies-equipment/thermometer-forehead-infrared-noncontact-6085215)

**9.0 Appendix 2: Full prototype circuit**

****

*Figure 22: Side-view of the combined prototype circuit*



*Figure 23: Top-view of the combined prototype circuit*

**10.0 Appendix 3: Full prototype code**





