SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Highly Commended

Computer Programming, Apps & Robotics

Year 7-8

Caleb Tang

Prince Alfred College

Government of South Australia
Department for Education

SASTA South Australian Science Teachers Association

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

Firetruck

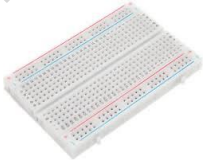
My project is a fully wireless automated Firetruck. Powered by solar panels and thermoelectric pads, the driver uses Bluetooth and Wi-Fi to control the truck. My project uses multiple sensors and cameras to alert the driver of what is going on around it and to track the conditions of the fire.

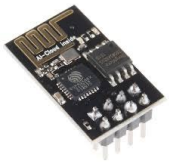
My project is designed to make firefighting safer. Over the bushfire season late last year and early this year, it was reported by the Parliament of Australia that 9 of the 33 total deaths from the fires were firemen. This could've easily been prevented through the usage of my project. By controlling and monitoring the truck from a safe distance away it means that there is a much lower risk of the firemen getting injured – a robot can be replaced but a person can never be brought back alive.

Not only are bushfires dangerous for the firefighters, but they are also extremely harmful to the environment. Approximately 400 million tons of carbon are released from fires in Australia every year, almost as much as Australia's annual human-caused emissions, and soot from bushfires can spread to places like New Zealand's glaciers, causing them to melt even quicker. Millions of animals need to travel over 20-30km to escape the heat and ferocity of the fires, and even if they do survive they will need to find completely new homes and food sources. This could also lead to a major loss of biodiversity, especially in Australia as many species are endemic to Australia. The use of my project would increase the rate at which fires could be put out, ultimately decreasing the social, economic and environmental impact of the fires.

Because my firetruck can be powered through solar power and thermoelectricity, two forms of energy that will be substantially abundant next to a fire, it means that the driver will rarely have to drive the truck back to refill, maximizing work time and efficiency. In addition, this also means that the firetruck will have the potential to work 24/7 with no rest as the driver can be easily switched. Some other features that make my firetruck stand out are that it uses Machine Learning to classify and detect people/animals and fire/smoke nearby. The AI will act as an extra pair of eyes for the driver because human errors and random errors such as lag and disconnections could cause the driver to miss things nearby. I trained my model with the Tensorflow Object Detection API and created a p5.js webpage to move a servo motor connected to an Arduino Leonardo when it detects something. Another feature that most firetrucks don't have is a remote monitoring system to monitor the conditions of the fire. I used runlinc to write a program that will display the water level in the truck, the weight of the water and how hot the area around the truck is in a table. It also alerts the driver if there are problems with the truck, e.g. if it's tipping over. This is an important feature to have as it is something that can't be easily detected through cameras, and if it isn't monitored properly it could end badly.

Although my project isn't fully refined yet and won't put a permanent stop to fires, it will make firefighting safer and easier to a large extent and is a significant improvement from what is traditionally used to fight fires. Moving forwards, I also hope to make my firetruck air based, i.e. helicopter/drone, so that it won't have to manoeuvre through the bush to reach the fire, but instead simply deploy water from above. I would also like to create a way to prevent fires from happening completely. This would be through slightly wetting areas likely to have fires every day, programming a robot to regularly mow the grass and rake the fields and tracking and reducing the amount of flammable fuels used in and around high-risk zones.

Components and Supplies	
Arduino Leonardo	Used in conjunction with the AI 
Arduino Uno	Used to control the car 
STEMSEL Board	Used to show Firetruck statistics 
Jumper Wires (Assorted)	Used to connect parts together 
Breadboard	Used with Bluetooth 
2 Wheel Car Base	Used to drive the car 
L298D Motor Driver H-Bridge	Used to control the motors 
HM-10 Bluetooth Module	Used to talk to the phone 
ESP8266 Wi-Fi Module	Used to connect the runlinc webpage to the STEMSEL board

	
Infrared Sensor	Used to stop the car when there are obstacles ahead 
Temperature Sensor	Used to track the temperature of the fire 
Light Sensor	Used to find water level in truck 
Weight Sensor	Used to find weight of water (L) 
LED's	Used to signal water level 

Necessary Tools and Machines	
Laptop with Webcam	Used to access AI webpage
Soldering Iron	Used to solder parts together 
Cables (Assorted)	Used to connect/power boards and upload code 
Batteries	Used to power parts

	
iPhone	Used to drive car 

Apps and Online Services	
Arduino IDE	Used to code car and AI sorter
Google Colab	Used Colab's cloud GPU to train AI
Tensorflow	Used Tensorflow model to train AI
Python	Language used in Colab notebook
P5.js	Used to make webpage for AI
runlinc	Used to make webpage for truck statistics
Google Drive	Used to sync everything together
Fritzing	Used to create Arduino schematics

Running my Project

To run my project, you will need to make sure that you have access to all the devices and parts required and are connected to a secure internet connection. To run the AI, after training your object detector paste the link onto the p5.js webpage while in present mode. Upload the Arduino code to the Leonardo board and give the browser webcam access. Then the AI will start looking for objects to classify. All you need to do to get the truck to drive is verify and upload the Arduino code before connecting your phone to the Bluetooth module. To get the statistics page to work simply connect the Wi-Fi module to the internet and load up the webpage. Upload your code and start running the project.

Code and Explanation

The code below is for the AI part of my truck. First is the .ipynb notebook script that I wrote to train my object detector. Although there were many different ways I could've trained my AI, I chose to use Google Colab as it was one of the only free online GPU's available. There were many different models that I could've used for my AI as well, but I chose the Tensorflow Object Detection API because of its speed to accuracy ratio and the large number of online resources available for it.

```
[ ] %tensorflow_version 1.x
```

```
[ ] import tensorflow
    print(tensorflow.__version__)
```

```
1.15.2
```

```
[ ] !pip install numpy==1.17.4
```

```
Requirement already satisfied: numpy==1.17.4 in /usr/local/lib/python3.6/dist-packages (1.17.4)
```

```
[ ] import tensorflow.compat.v1 as tf
    import cv2

    from PIL import Image
    from collections import namedtuple, OrderedDict

    from google.colab import files
```

Here I download the Tensorflow model that contains the Object Detection API. This is the API that I will use to train my model. After that I compile my protocol buffers, which are used to transfer data from the cloud to Google Colab's GPU.

```
[ ] # downloads the models
    %cd /gdrive/'My Drive'/object_detection
    !git clone --q https://github.com/tensorflow/models.git
```

```
/gdrive/My Drive/object_detection
fatal: destination path 'models' already exists and is not an empty directory.
```

```
[ ] %cd /gdrive/'My Drive'/object_detection/models/research
    !protoc object_detection/protos/*.proto --python_out=.

    # exports the PYTHONPATH environment variable with the reasearch and slim folders' paths
    os.environ['PYTHONPATH'] += ':/gdrive/My Drive/object_detection/models/research/:/gdrive/My Drive/object_detection/models/research/slim/'
```

```
/gdrive/My Drive/object_detection/models/research
```

```
[ ] # testing the model builder
    !python3 object_detection/builders/model_builder_test.py
```

Next I run a quick test to confirm that the model builder is working properly.

```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

Running tests under Python 3.6.9: /usr/bin/python3
[ RUN      ] ModelBuilderTest.test_create_experimental_model
[       OK ] ModelBuilderTest.test_create_experimental_model
[ RUN      ] ModelBuilderTest.test_create_faster_rcnn_model_from_config_with_example_miner
[       OK ] ModelBuilderTest.test_create_faster_rcnn_model_from_config_with_example_miner
[ RUN      ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_with_matmul
[       OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_with_matmul
[ RUN      ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul
[       OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_faster_rcnn_without_matmul
[ RUN      ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
[       OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_with_matmul
[ RUN      ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_without_matmul
[       OK ] ModelBuilderTest.test_create_faster_rcnn_models_from_config_mask_rcnn_without_matmul
[ RUN      ] ModelBuilderTest.test_create_rfcn_model_from_config
[       OK ] ModelBuilderTest.test_create_rfcn_model_from_config
[ RUN      ] ModelBuilderTest.test_create_ssd_fpn_model_from_config
[       OK ] ModelBuilderTest.test_create_ssd_fpn_model_from_config
[ RUN      ] ModelBuilderTest.test_create_ssd_models_from_config
[       OK ] ModelBuilderTest.test_create_ssd_models_from_config
[ RUN      ] ModelBuilderTest.test_invalid_faster_rcnn_batchnorm_update
[       OK ] ModelBuilderTest.test_invalid_faster_rcnn_batchnorm_update
[ RUN      ] ModelBuilderTest.test_invalid_first_stage_nms_iou_threshold
[       OK ] ModelBuilderTest.test_invalid_first_stage_nms_iou_threshold
[ RUN      ] ModelBuilderTest.test_invalid_model_config_proto
[       OK ] ModelBuilderTest.test_invalid_model_config_proto
[ RUN      ] ModelBuilderTest.test_invalid_second_stage_batch_size
[       OK ] ModelBuilderTest.test_invalid_second_stage_batch_size
[ RUN      ] ModelBuilderTest.test_session
[  SKIPPED ] ModelBuilderTest.test_session
[ RUN      ] ModelBuilderTest.test_unknown_faster_rcnn_feature_extractor
[       OK ] ModelBuilderTest.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTest.test_unknown_meta_architecture
[       OK ] ModelBuilderTest.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTest.test_unknown_ssd_feature_extractor
[       OK ] ModelBuilderTest.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
Ran 17 tests in 0.168s

OK (skipped=1)
```

```
[ ]  #adjusted from: https://github.com/datitran/raccoon_dataset
     from object_detection.utils import dataset_util
     %cd /gdrive/My Drive/object_detection/models/research

     #change this to the base directory where your data/ is
     data_base_url = '/gdrive/My Drive/object_detection/data/'

     #location of images
     image_dir = data_base_url +'images/'

     def class_text_to_int(row_label):
         if row_label == 'cardboard':
             return 1
         if row_label == 'glass':
             return 2
         if row_label == 'metal':
             return 3
         if row_label == 'paper':
             return 4
         if row_label == 'plastic':
             return 5
         else:
             None
```

Here I start to preprocess the images and labels in preparation for training the AI. After collecting my pictures and annotating them in LabelIMG I convert them from a couple hundred .xml files to two .csv files; one .csv file for the train labels (used to train the model) and one .csv file for the test labels (to test the model). Once I have the .csv files I can use them to create a labelmap.pbtxt file. This specifies the different classes that I will classifv.

```
[ ]  def split(df, group):
       data = namedtuple('data', ['filename', 'object'])
       gb = df.groupby(group)
       return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]

     def create_tf_example(group, path):
       with tf.io.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
         encoded_jpg = fid.read()
       encoded_jpg_io = io.BytesIO(encoded_jpg)
       image = Image.open(encoded_jpg_io)
       width, height = image.size
       filename = group.filename.encode('utf8')
       image_format = b'jpg'
       xmins = []
       xmaxs = []
       ymins = []
       ymaxs = []
       classes_text = []
       classes = []

       for index, row in group.object.iterrows():
         xmins.append(row['xmin'] / width)
         xmaxs.append(row['xmax'] / width)
         ymins.append(row['ymin'] / height)
         ymaxs.append(row['ymax'] / height)
         classes_text.append(row['class'].encode('utf8'))
         classes.append(class_text_to_int(row['class']))
```

I got the majority of this code from github, but essentially what it does is it uses the .csv files to generate two TFRecord files; one for training and one for testing. The TFRecord files will be used to run the epochs on as the files store all of the data and are quite small, maximizing the training efficiency.


```
[ ]    tf_example = tf.train.Example(features=tf.train.Features(feature={
         'image/height': dataset_util.int64_feature(height),
         'image/width': dataset_util.int64_feature(width),
         'image/filename': dataset_util.bytes_feature(filename),
         'image/source_id': dataset_util.bytes_feature(filename),
         'image/encoded': dataset_util.bytes_feature(encoded_jpg),
         'image/format': dataset_util.bytes_feature(image_format),
         'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
         'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
         'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
         'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
         'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
         'image/object/class/label': dataset_util.int64_list_feature(classes),
       }))
       return tf_example
     #creates tfrecord for both csv's
     for csv in ['train_labels', 'test_labels']:
       writer = tf.io.TFRecordWriter(data_base_url + csv + '.record')
       path = os.path.join(image_dir)
       examples = pd.read_csv(data_base_url + csv + '.csv')
       grouped = split(examples, 'filename')
       for group in grouped:
         tf_example = create_tf_example(group, path)
         writer.write(tf_example.SerializeToString())

       writer.close()
       output_path = os.path.join(os.getcwd(), data_base_url + csv + '.record')
       print('Successfully created the TFRecords: {}'.format(data_base_url +csv + '.record'))
```

```
/gdrive/My Drive/object_detection/models/research
Successfully created the TFRecords: /gdrive/My Drive/object_detection/data/train_labels.record
Successfully created the TFRecords: /gdrive/My Drive/object_detection/data/test_labels.record
```

```
[ ] # Some models to train on
    MODELS_CONFIG = {
        'ssd_mobilenet_v2': {
            'model_name': 'ssd_mobilenet_v2_coco_2018_03_29',
            'pipeline_file': 'ssd_mobilenet_v2_coco.config',
        },
        'faster_rcnn_inception_v2': {
            'model_name': 'faster_rcnn_inception_v2_coco_2018_01_28',
        },
    }

    # Select a model from `MODELS_CONFIG`.
    # I chose ssd_mobilenet_v2 for this project, you could choose any
    selected_model = 'ssd_mobilenet_v2'
```

I didn't write all of this code, but I modified it to suit my needs. This script is configuring the training pipeline. I also add the path to the TFRecord files and the .pbtxt file. Here it is selecting the kind of configuration model that I want to train my AI on. I used 'ssd_mobilenet_v2'.

```
[ ] #the distination folder where the model will be saved
    #change this if you have a different working dir
    DEST_DIR = '/gdrive/My Drive/object_detection/models/research/pretrained_model'

    # Name of the object detection model to use.
    MODEL = MODELS_CONFIG[selected_model]['model_name']

    #selecting the model
    MODEL_FILE = MODEL + '.tar.gz'

    #creating the downlaod link for the model selected
    DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

    #checks if the model has already been downloaded, download it otherwise
    if not (os.path.exists(MODEL_FILE)):
        urllib.request.urlretrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

    #unzipping the model and extracting its content
    tar = tarfile.open(MODEL_FILE)
    tar.extractall()
    tar.close()

    # creating an output file to save the model while training
    os.remove(MODEL_FILE)
    if (os.path.exists(DEST_DIR)):
        shutil.rmtree(DEST_DIR)
    os.rename(MODEL, DEST_DIR)
```

```
[ ]
    #path to the config file
    %%writefile object_detection/samples/configs/ssd_mobilenet_v2_coco.config

    # paste the content of the config file in the same cell here.
    # SSD with Mobilenet v2 configuration for MSCOCO Dataset.
    # Users should configure the fine_tune_checkpoint field in the train config as
    # well as the label_map_path and input_path fields in the train_input_reader and
    # eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
    # should be configured.

    model {
      ssd {
        num_classes: 6
        box_coder {
          faster_rcnn_box_coder {
            y_scale: 10.0
            x_scale: 10.0
            height_scale: 5.0
            width_scale: 5.0
          }
        }
        matcher {
          argmax_matcher {
            matched_threshold: 0.5
            unmatched_threshold: 0.5
            ignore_thresholds: false
            negatives_lower_than_unmatched: true
            force_match_for_each_row: true
          }
        }
        similarity_calculator {
          iou_similarity {
          }
        }
```

```
[ ]       anchor_generator {
            ssd_anchor_generator {
              num_layers: 6
              min_scale: 0.2
              max_scale: 0.95
              aspect_ratios: 1.0
              aspect_ratios: 2.0
              aspect_ratios: 0.5
              aspect_ratios: 3.0
              aspect_ratios: 0.3333
            }
          }
          image_resizer {
            fixed_shape_resizer {
              height: 300
              width: 300
            }
          }
          box_predictor {
            convolutional_box_predictor {
              min_depth: 0
              max_depth: 0
              num_layers_before_predictor: 0
              use_dropout: true
              dropout_keep_probability: 0.8
              kernel_size: 1
              box_code_size: 4
              apply_sigmoid_to_scores: false
              conv_hyperparams {
                activation: RELU_6,
                regularizer {
                  l2_regularizer {
                    weight: 0.00004
                  }
                }
[ ]             initializer {
                  truncated_normal_initializer {
                    stddev: 0.03
                    mean: 0.0
                  }
                }
                batch_norm {
                  train: true,
                  scale: true,
                  center: true,
                  decay: 0.9997,
                  epsilon: 0.001,
                }
              }
            }
          }
          feature_extractor {
            type: 'ssd_mobilenet_v2'
            min_depth: 16
            depth_multiplier: 1.0
            conv_hyperparams {
              activation: RELU_6,
              regularizer {
                l2_regularizer {
                  weight: 0.00004
                }
              }
              initializer {
                truncated_normal_initializer {
                  stddev: 0.03
                  mean: 0.0
                }
              }
              batch_norm {
                train: true,
```

```
[ ]           scale: true,
              center: true,
              decay: 0.9997,
              epsilon: 0.001,
            }
          }
        }
        loss {
          classification_loss {
            weighted_sigmoid {
            }
          }
          localization_loss {
            weighted_smooth_l1 {
            }
          }
          hard_example_miner {
            num_hard_examples: 3000
            iou_threshold: 0.99
            loss_type: CLASSIFICATION
            max_negatives_per_positive: 3
            min_negatives_per_image: 3
          }
          classification_weight: 1.0
          localization_weight: 1.0
        }
        normalize_loss_by_num_matches: true
        post_processing {
          batch_non_max_suppression {
            score_threshold: 1e-8
            iou_threshold: 0.6
            max_detections_per_class: 100
            max_total_detections: 100
          }
          score_converter: SIGMOID
[ ]     }
      }
    }

    train_config: {
      batch_size: 24
      optimizer {
        rms_prop_optimizer: {
          learning_rate: {
            exponential_decay_learning_rate {
              initial_learning_rate: 0.004
              decay_steps: 800720
              decay_factor: 0.95
            }
          }
          momentum_optimizer_value: 0.9
          decay: 0.9
          epsilon: 1.0
        }
      }
      fine_tune_checkpoint: "/gdrive/My Drive/object_detection/models/research/pretrained_model/model.ckpt"
      fine_tune_checkpoint_type:  "detection"
      # Note: The below line limits the training process to 200K steps, which we
      # empirically found to be sufficient enough to train the pets dataset. This
      # effectively bypasses the learning rate schedule (the learning rate will
      # never decay). Remove the below line to train indefinitely.
      num_steps: 20000
      data_augmentation_options {
        random_horizontal_flip {
        }
      }
      data_augmentation_options {
        ssd_random_crop {
        }
      }
```

Here I make my batch size 24 and I train my AI with 20,000 steps. I also augment and adjust some of my images to reduce errors. Although this means it will take longer to train it will come out more accurate and work better. The directory below is where I will save the model at each checkpoint while training.

```
[ ] }

    train_input_reader: {
      tf_record_input_reader {
        input_path: "/gdrive/My Drive/object_detection/data/train_labels.record"
      }
      label_map_path: "/gdrive/My Drive/object_detection/data/label_map.pbtxt"
    }

    eval_config: {
      num_examples: 8000
      # the number of images to disply in Tensorboard while training
      num_visualizations: 20
    }

    eval_input_reader: {
      tf_record_input_reader {
        input_path: "/gdrive/My Drive/object_detection/data/test_labels.record"
      }
      label_map_path: "/gdrive/My Drive/object_detection/data/label_map.pbtxt"
      shuffle: false
      num_readers: 1
    }
```

```
Overwriting object_detection/samples/configs/ssd_mobilenet_v2_coco.config
```

```
[ ] # where the model will be saved at each checkpoint while training
    model_dir = 'training/'

    # Optionally: remove content in output model directory to fresh start.
    !rm -rf {model_dir}
    os.makedirs(model_dir, exist_ok=True)
```

```
[ ] !wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
    !unzip -o ngrok-stable-linux-amd64.zip
```

```
--2020-04-20 23:11:27--  https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
Resolving bin.equinox.io (bin.equinox.io)... 34.237.57.234, 34.192.108.200, 34.226.171.201, ...
Connecting to bin.equinox.io (bin.equinox.io)|34.237.57.234|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13773305 (13M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.zip.11'

ngrok-stable-linux- 100%[===================>]  13.13M  12.5MB/s    in 1.0s

2020-04-20 23:11:28 (12.5 MB/s) - 'ngrok-stable-linux-amd64.zip.11' saved [13773305/13773305]

Archive:  ngrok-stable-linux-amd64.zip
  inflating: ngrok
```

This step is downloading and unzipping Tensorboard. Tensorboard is a package that allows you to visually track the AI's progress while it is training. It creates multiple graphs that update in real time to show things such as the AI's total loss, the epochs, etc.

```
[ ] #the logs that are created while training
    LOG_DIR = 'training/'
    get_ipython().system_raw(
        'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
        .format(LOG_DIR)
    )
    get_ipython().system_raw('./ngrok http 6006 &')
    #The link to tensorboard.
    #works after the training starts.
    !curl -s http://localhost:4040/api/tunnels | python3 -c \
        "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

```
http://72dbb005.ngrok.io
```

```
[ ] !python3 object_detection/model_main.py \
        --pipeline_config_path=/gdrive/My\ Drive/object_detection/models/research/object_detection/samples/configs/ssd_mobilenet_v2_coco.config \
        --model_dir=training/
```

```
[ ] output_directory = './fine_tuned_model'

    lst = os.listdir('training')
    lst = [l for l in lst if 'model.ckpt-' in l and '.meta' in l]
    steps=np.array([int(re.findall('\d+', l)[0]) for l in lst])
    last_model = lst[steps.argmax()].replace('.meta', '')

    last_model_path = os.path.join('training', last_model)


    !python /gdrive/'My Drive'/object_detection/models/research/object_detection/export_inference_graph.py \
        --input_type=image_tensor \
        --pipeline_config_path=/gdrive/My\ Drive/object_detection/models/research/object_detection/samples/configs/ssd_mobilenet_v2_coco.config \
        --output_directory={output_directory} \
        --trained_checkpoint_prefix={last_model_path}
```

This portion of the code is actually training the model. This step took over 20 hours for me as I was doing it on my laptop's CPU. Next time if it was trained on a GPU it would be much quicker.

```
[ ]  #downloads the frozen model that is needed for inference
     # output_directory = 'fine_tuned_model' dir specified above.
     files.download(output_directory + '/frozen_inference_graph.pb')
```

```
[ ]  #downlaod the label map
     # we specified 'data_base_url' above. It directs to
     # 'object_detection/data/' folder.
     files.download(data_base_url + '/label_map.pbtxt')
```

After being trained, the model is exported and uploaded to the cloud to generate the link for p5.js to run.

The next set of code is done on p5.js. p5.js is a JavaScript library that is often used to create webpages. I chose to use p5.js as it is online and free, easy to run and quick to operate.

Sketch.js

```
class PhotoGrid {
    constructor(isLeft) {
      this.images = [];
      console.log(this.images.length);
      if (isLeft) {
        this.x = width / 2 - 480;
      } else {
        this.x = width / 2 + 300;
      }

      this.y = height / 2.5;
      this.numRows = 3;
      this.numCols = 2;
      this.imageSize = 120;
      this.padding = 20;
    }

    addImage(img) {
      this.images.push(img);
      if (this.images.length > 9) {
        this.images.shift();

      }
    }

    render() {

      for (let i = 0; i < this.images.length; i++) {

        let currImage = this.images[i];
        let row = i % 3;
        let col = int(i / 3);
        fill(255);
        noStroke();
        rectMode(CORNER);
        rect(this.x + (this.imageSize + this.padding) * col, this.y + (this.imageSize + this.padding) *
row,
          this.imageSize, this.imageSize, 3, 3, 3, 3);

        image(currImage, this.x + (this.imageSize + this.padding) * col + 5, this.y + (this.imageSize +
this.padding) * row + 5,
          this.imageSize - 10, this.imageSize - 10);
      }
    }


  }


  class DropDown {
    constructor(isLeft, classList) {
      this.isClicked = false;
      this.width =
        this.height = 3;
      this.x = 0;
      this.y = 0;

      this.class = null;
    }

    render() {

    }


  }

  function debounce(func, wait, immediate) {
    var timeout;
    return function() {
      var context = this,
        args = arguments;
      var later = function() {
        timeout = null;
        if (!immediate) func.apply(context, args);
      };
      var callNow = immediate && !timeout;
      clearTimeout(timeout);
      timeout = setTimeout(later, wait);
      if (callNow) func.apply(context, args);
    };
  }
```

The code starts off by creating the class 'PhotoGrid'. Here it is saying where the webcam image will go by giving the x and y coordinates.

The 'render()' function here tells the browser to show the output from the browser. Below it also

The class 'DropDown' basically tells the browser to ask for permission to access the camera. Once accepted it will start streaming video from the webcam. The function 'debounce()' below tells the browser to not turn

Lines 84 to 107 tells the browser what colour the text under class 'Splash' will be and where they are positioned.

```
class Splash {

  constructor(isLeft) {
    if (isLeft) {
      this.x = width / 2 + 314;
    } else {
      this.x = width / 2 - 314
    }
    this.y = height / 3.3;
    this.color = color(147, 229, 21);
    this.isExploding = false;
    this.isInbetweenUpdates = false;
    this.explosionRadius = 100;
    this.explosionIndex = 0;
    this.numRadius = 4;
    this.radiusOffset = 10;
    this.width = 243;
    this.height = 53;
  }

  updatePosition(x, y) {
    this.x = x;
    this.y = y;
  }
```

Lines 110 to 144 basically states when and how often the 'Splash' text and the webcam will be updated.

```
  trigger() {
    this.isExploding = true;
  }

  updateIndex() {
    this.explosionIndex++;
    this.isInbetweenUpdates = false;
  }

  render() {
    if (!this.isExploding) {
      fill(this.color);
      // rect(this.x, this.y, this.width, this.height);
    } else {
      noFill();
      strokeWeight(3);
      stroke(this.color);
      rect(this.x, this.y, this.width + (this.radiusOffset * this.explosionIndex),
        this.height + (this.radiusOffset * this.explosionIndex), 9, 9, 9, 9);
    }

    if (this.isExploding && !this.isInbetweenUpdates) {
      setTimeout(() => {
        this.updateIndex()
      }, 100);
      this.isInbetweenUpdates = true;
    }

    if (this.explosionIndex >= this.numRadius) {
      this.isExploding = false;
      this.isInbetweenUpdates = false;
      this.explosionIndex = 0;
    }

  }


}
```

Lines 149 to 155 tells the webpage how big the classification bar is and where it is located on the page.

```
class ClassificationBar {
  constructor() {
    this.width = min(width / 4, 341);
    this.height = 28;
    this.x = width / 2;
    this.y = height / 3.3;
    this.radius = 5;

    this.classificationLeft = 0;
    this.classificationMaxWidth = this.width / 2;
    this.classificationRight = 0.0;
    this.hasSetTimeout = false;
  }


  updateClassification(results) {
    // console.log(results);
    const class1 = results.filter(objs => {
```

```
      if (objs.label === labels[0]) {
        return objs;
      }
    });

    const class2 = results.filter(objs => {

      if (objs.label === labels[1]) {
        return objs;
      }
    });

    this.classificationLeft = map(class1[0].confidence, 0, 1.0, 0, this.classificationMaxWidth);
    this.classificationRight = map(class2[0].confidence, 0, 1.0, 0, this.classificationMaxWidth);

    let view = new Uint8Array(1);

    if (class1[0].confidence > 0.90) {
      view[0] = 1;
      try {
        port.send(view);
        shouldFreezeFrame = true;
        splashLeft.trigger();

        isLeftPic = false;
      } catch (e) {}
    } else if (class2[0].confidence > 0.90) {
      view[0] = 2;
      try {
        port.send(view);
        shouldFreezeFrame = true;
        splashRight.trigger();
        isLeftPic = true;
      } catch (e) {}


    }
  }

  render() {
    //Draw Background rectangle
    rectMode(CENTER);
    fill('rgba(174, 203, 250, 0.4)');
    stroke(255);
    strokeWeight(5);
    rect(this.x, this.y, this.width, this.height, this.radius, this.radius, this.radius, this.radius);
    noStroke();


    fill('#19ce1f');
    rect(this.x + this.classificationLeft / 2, this.y, this.classificationLeft, this.height, this.radius,
this.radius, this.radius, this.radius);
    rect(this.x - this.classificationRight / 2, this.y, this.classificationRight, this.height,
this.radius, this.radius, this.radius, this.radius);
    stroke(0);
    strokeWeight(7);
    strokeCap(ROUND);
    line(this.x, this.y - this.height / 2, this.x, this.y + this.height / 2);

  }
}

class ClassInput {

  constructor(isLeft) {
    this.width = 243;
    this.height = 53;
    this.radius = 9;
    this.textLineOffset = 40;
    this.isLeft = isLeft;
    this.hoverOne = false;
    this.hoverTwo = false;
    this.hoverThree = true;
    if (isLeft === true) {
      this.x = width / 2 + 314;
    } else {
      this.x = width / 2 - 314;
    }

    this.y = height / 3.3;
    this.isActive = false;
    this.currentValue = null;

  }
```

Lines 164 to 177 updates the classification bar constantly according to what is streamed from the webcam. The browser will send out either [0] or [1] according to what it detects, and also give a confidence rating of how confident it is of what it sees.

Here it is saying that if the confidence rating for either class1 or class2 is higher than 90% it will send out a signal of either 1 or 2 that will be received by the Arduino Leonardo.

Lines 207 to 226 tell the webpage what colour the background should be and draws it up. The hex colour and the RGB colour is the colour of the background

This part if the code creates the 'load model' button. It gives the browser the x and y coordinates of the button and tells it how big they should be. It also creates the space for you to paste the trained model's link. Then it will send out a signal once it has been clicked.

```
    onClick(x, y) {

      const leftBound = this.x - this.width / 2;
      const rightBound = this.x + this.width / 2;
      const bottomBound = this.y + this.height / 2;
      const topBound = this.y - this.height / 2;
      const isInside = (x >= leftBound && x <= rightBound && y <= bottomBound && y >= topBound);

      if (isInside) {
        this.isActive = !this.isActive;

      }

    }

    onHover(x, y) {
      this.detectZone(x, y);

    }
    detectZone(x, y) {
      const leftBound = this.x - this.width / 2;
      const rightBound = this.x + this.width / 2;

      const zoneOneBottom = this.y + this.height / 2;
      const zoneOneTop = this.y - this.height / 2;

      if (x >= leftBound && x <= rightBound && y <= zoneOneTop && y >= zoneOneBottom) {
        this.hoverOne = true;
        this.hoverTwo = false;
        this.hoverThree = false;
        return 1;
      }
    }
    render() {
      if (isModelLoaded) {
        fill(255);
        rectMode(CENTER);
        noStroke();
        textFont(poppinsBold);
        textSize(24);
        // if (!this.isActive) {

          rect(this.x, this.y, this.width, this.height, this.radius, this.radius, this.radius,
this.radius);

        // } else {
        //   rect(this.x, this.y + this.textLineOffset, this.width, this.height + this.textLineOffset * 2,
this.radius, this.radius, this.radius, this.radius);
        //   if (this.hoverOne) {
        //     fill('rgba(154,160,166, 0.2)');
        //     rect(this.x, this.y, this.width, this.height, this.radius, this.radius, 0, 0);
        //   } else if (this.hoverTwo) {
        //     fill('rgba(154, 160, 166, 0.2)');
        //     rect(this.x, this.y + this.textLineOffset, this.width, this.height - 10, 0, 0, 0, 0);
        //   } else if (this.hoverThree) {
        //     fill('rgba(154, 160, 166, 0.2)');
        //     rect(this.x, this.y + this.textLineOffset * 2 + 6, this.width, this.height - 11, 0, 0,
this.radius, this.radius);
        //   }
        // }
        if (labels.length >= 2) {
          fill('#19ce1f');
          if (this.isLeft) {
            textAlign(LEFT, CENTER);
            text(labels[0], this.x - this.width / 2 + 10, this.y - 4);
  //           if (this.isActive) {
  //             text(labels[1], this.x - this.width / 2 + 10, this.y + this.textLineOffset);
  //             text(labels[2], this.x - this.width / 2 + 10, this.y + this.textLineOffset * 2);

  //           }
            image(pencil, this.x - this.width / 2 + 200, this.y - this.height / 2 + 10, pencil.width / 2,
pencil.height / 2);
          } else {
            textAlign(RIGHT, CENTER);
            // if (labels.length > 2) {
              text(labels[1], this.x + this.width / 2 - 13, this.y - 4);
            // }

  //           if (this.isActive) {
  //             text(labels[1], this.x + this.width / 2 - 13, this.y + this.textLineOffset);
  //             text(labels[2], this.x + this.width / 2 - 13, this.y + this.textLineOffset * 2);
  //           }
            image(pencil, this.x + this.width / 2 - 235, this.y - this.height / 2 + 10, pencil.width / 2,
pencil.height / 2);
```

Here it checks if the button has been clicked or not, and tells the webpage what the button will say, what colour it is and what font it is in.

```
      }
    }


  }
 }
}

// Classifier Variable
let classifier;
let input;
// Model URL
let imageModel = 'https://teachablemachine.withgoogle.com/models/9L4-MDs0/';

// Video
let video;
let videoSize;
let classificationIndicator;

let leftGrid;
let leftAdd;
let rightGrid;
let rightAdd;

let isLeftPic;

let leftClassSelector;
let rightClassSelector;

let cameraBorder;
let title;
let splashLeft;
let splashRight;
let selectPic;

let editCode;
let connect;
let group;
let pencil;
// Darker BG
// let bgColor = '#63e446';
// Lighter Bg
let bgColor = '#bce446';
let port;
let shouldFreezeFrame;
let modelInput;
let loadModel;
let labels = [];
let isLeftClassSelected = false;
let isRightClassSelected = false;

let poppinsRegular;
let poppinsBold;
let hasSetPauseTimer;
// To store the classification
let label = "";
let isModelLoaded = false;
let enteredText = ""
// // Load the model first
// function preload() {
//   classifier = ml5.imageClassifier(imageModel + 'model.json');
// }

function myInputEvent() {
  enteredText = this.value();
}

function setup() {
  createCanvas(window.innerWidth, window.innerHeight);
  // Create the video
  videoSize = 250;
  video = createCapture(VIDEO);
  video.hide();

  cameraBorder = loadImage('camera_border.png');
  title = loadImage('title.png');
  group = loadImage('Group 61.png');

  loadModel = new Clickable();

  loadModel.resize(145, 40);

  loadModel.locate(300, 15);
  loadModel.strokeWeight = 0;
  loadModel.color = bgColor;
```

This is the model URL to the trained model where the machine learning model will be hosted. It stores the weights and the .json files for the object detector.

Lines 345 to 388 define the let functions. The let functions declare all of the different variables. It starts off with declaring all the terms for getting the video running before declaring the background colours and fonts.

Here the webpage loads the model and starts up the video. The video dimensions are determined as well before some pictures are imported from the sketch files.

The Clickable() function works with the p5.clickable.min.js file to create the 'LOAD MODEL' button. It is dimensioned, sized and coloured before being told what it will do one clicked. Here it says that if there are 2 or less different classes it will come up with an alert. Otherwise, it will load the model and start classifying the video.

```
loadModel.text = 'LOAD MODEL';
loadModel.textSize = 18;
loadModel.textColor = '#19ce1f';
loadModel.onPress = () => {
  try {
    console.log(enteredText + 'metadata.json');
    classifier = ml5.imageClassifier(enteredText + 'model.json');

    httpGet(enteredText + 'metadata.json', 'json', false, (response) => {
      if (response.labels.length <= 2) {
        alert("Train a model with at least three classes: one for each type of object you want to sort,
and one for the empty sorter");
      } else {
        labels = response.labels;
        isModelLoaded = true;
        classifyVideo();
      }

    }, (error) => alert("invalid TM2 url"));
  } catch (e) {
    loadModel.text = 'INVALID URL';
  }
  if (labels.length > 1) {
    loadModel.text = 'MODEL LOADED';
    setTimeout(() => {
      loadModel.text = 'REFRESH MODEL'
    }, 3000);
  }
}

leftGrid = new PhotoGrid(true);
pencil = loadImage('pencil_icon.png');
classificationIndicator = new ClassificationBar();
leftClassSelector = new ClassInput(true);
rightClassSelector = new ClassInput(false);
splashRight = new Splash(false);
splashLeft = new Splash(true);
rightGrid = new PhotoGrid(false);
poppinsRegular = loadFont('Poppins-Regular.ttf');
poppinsBold = loadFont('Poppins-Bold.ttf');

loadModel.textFont = poppinsRegular;
shouldFreezeFrame = false;
hasSetPauseTimer = false;
var serial = {};

modelInput = createInput();
modelInput.input(myInputEvent);
// modelInput.style('position', 'absolute');
// modelInput.style('z-index', 10);
modelInput.position(20, 20);
modelInput.style('height', '35px');
modelInput.style('width', '267px');
modelInput.style('border-width', '0px');
modelInput.style('border-radius', '4px 4px 0px 0px');
modelInput.style('border-bottom', '2px solid #19ce1f');
modelInput.style('font-family', 'Poppins');
modelInput.style('font-size', '16px');
modelInput.style('padding-left', '5px');
modelInput.style('color', '#19ce1f');
modelInput.attribute('placeholder', "Paste model link here");


connect = createButton('CONNECT ARDUINO');
connect.position(width - 200, 20);
connect.id("connect");
connect.style('height', '40px');
connect.style('border-width', '0px');
connect.style('background-color', bgColor);
connect.style('font-family', 'Poppins');
connect.style('font-size', '18px');
connect.style('width', '200px');
connect.style('color', '#19ce1f');
leftAdd = debounce(() => {
  leftGrid.addImage(selectPic)
}, 500, true);
rightAdd = debounce(() => {
  rightGrid.addImage(selectPic)
}, 500, true);

editCode = createA('https://editor.p5js.org/ctang21/sketches/ttnyj7hOV', 'EDIT CODE', '_blank');
editCode.position(width - 110, height - 40);
editCode.style('height', '40px');
editCode.style('border-width', '0px');
editCode.style('background-color', bgColor);
```

Here it will come up with an alert if the trained model link is incorrect if it doesn't work. If it is fine, it will say 'MODEL LOADED' and give the option to 'REFRESH MODEL' in case you updated it.
Below the 'pencil_icon.png' image is loaded onto the classification bar and the text, fonts and colours are also imported.

This section of the code creates the textbox where you can paste the trained model's link. After that a button is created to go in the top right corner of the page that says 'CONNECT ARDUINO'. It is also coloured and sized.
Below that a button that says 'EDIT CODE' is created. If clicked, it will take you to the p5.js sketch where you can edit the code.

```
    editCode.style('font-family', 'Poppins');
    editCode.style('font-size', '18px');
    editCode.style('width', '200px');
    editCode.style('color', '#19ce1f');

    // Start classifying
    if (isModelLoaded) {
      classifyVideo();
    }
  }

  function draw() {
    //   Darker BG
    if (width > 700) {
      background(bgColor);
      video.get();
      //   Darker BG
      // background('#e8f0fe');
      if (shouldFreezeFrame && !hasSetPauseTimer) {
        video.pause();
        selectPic = video.get(150, 0, videoSize / 1.6, videoSize / 1.6);
        if (isLeftPic) {
          leftAdd();
        } else {
          rightAdd();
        }
        setTimeout(() => {

          video.play();
          hasSetPauseTimer = false;
          shouldFreezeFrame = false

        }, 2000);
      }

      noStroke();
      textFont(poppinsBold);
        textAlign(CENTER, CENTER);
        textSize(14);
        text("enable webcam access", width / 2, height / 1.6);
        text("and refresh page to use", width / 2, height / 1.5);
      image(title, width / 2 - title.width / 5, 0, title.width / 2.5, title.height / 2.5);
      image(video, width / 2 - videoSize / 2, height / 1.6 - videoSize / 2, videoSize, videoSize, 150, 0,
videoSize * 1.5, videoSize * 1.5);
      image(cameraBorder, width / 2 - videoSize / 2 - 3, height / 1.6 - videoSize / 2 - 3, videoSize + 6,
videoSize + 6);

      // // image(connect, width - connect.width - 20, 20);
      // image(group, 20, 20);

      // rectMode(CENTER);
      // noFill();
      // stroke(255);
      // strokeWeight(6);
      // rect(width / 2, height / 2, videoSize, videoSize);
      leftGrid.render();
      rightGrid.render();
      rectMode(CORNER);
      loadModel.draw();
      classificationIndicator.render();
      leftClassSelector.render();
      rightClassSelector.render();
      splashLeft.render();
      splashRight.render();
    } else {
      noStroke();

      text("expand page or ", width / 2, height / 1.6);
      text("load on a computer to use", width / 2, height / 1.5);
    }
  }

  // Get a prediction for the current video frame
  function classifyVideo() {
    classifier.classify(video, gotResult);
    // classifier.classify(video, () => {});
  }

  // When we get a result
  function gotResult(error, results) {
    // If there is an error
    if (error) {
      console.error(error);
      return;
    }
```

This portion of the code tells the pictures taken from the webcam stream once they have been classified what should happen to them and where to go.

Here it says what will happen if webcam access isn't granted; the webcam will ask for control of the webpage before it tells you to refresh the page to see the webcam working.

If the webpage is too small the camera will not be accessed, and so here it tells you to expand the page.

This is where the video is finally classified. It will make a prediction of what it is and give a percentage of certainty before telling you what it classified it as. Once you receive a result the webcam will start classifying again.

```
  // The results are in an array ordered by confidence.
  // console.log(results[0]);
  classificationIndicator.updateClassification(results);

  label = results[0].label;
  // Classify again!
  classifyVideo();
}

function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
  const leftPhotos = leftGrid.images;
  const rightPhotos = rightGrid.images;
  leftGrid = new PhotoGrid(true);
  rightGrid = new PhotoGrid(false);
  leftGrid.images = leftPhotos;
  rightGrid.images = rightPhotos;
  classificationIndicator = new ClassificationBar();
  leftClassSelector = new ClassInput(true);
  rightClassSelector = new ClassInput(false);
  splashRight = new Splash(false);
  splashLeft = new Splash(true);
  loadModel = new Clickable();
  connect.position(width - 200, 20);
  loadModel.resize(145, 40);
  loadModel.locate(300, 15);
  loadModel.strokeWeight = 0;
  loadModel.color = '#19ce1f';
  loadModel.text = 'LOAD MODEL';
  loadModel.textSize = 18;
  loadModel.textColor = '#19ce1f';
  loadModel.onPress = () => {
    loadModel.text = 'MODEL LOADED';
    setTimeout(() => {
      loadModel.text = 'REFRESH MODEL'
    }, 3000);
  }
  // connect.textFont = poppinsRegular;
  loadModel.textFont = poppinsRegular;
}

function mousePressed() {
  leftClassSelector.onClick(mouseX, mouseY);
  rightClassSelector.onClick(mouseX, mouseY);
}

function mouseMoved() {
  leftClassSelector.onHover(mouseX, mouseY);
  rightClassSelector.onHover(mouseX, mouseY);
}
```

The windowResized() function tells the webpage how big everything should be if the browser is extended. This is so that the ratio of the size of the text/buttons/shapes to the size of the webpage stays the same. It also makes it so that the text/buttons/shapes are evenly distributed throughout the page. Here it updates the video, the classification bar and the buttons.

Index.html

```
<!DOCTYPE html>
<html>

<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/addons/p5.dom.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/addons/p5.sound.min.js"></script>
  <script src="https://unpkg.com/ml5@0.4.1/dist/ml5.min.js"></script>
  <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap" rel="stylesheet">

  <link rel="stylesheet" type="text/css" href="style.css">
  <link rel="stylesheet" type="text/css" href="purejscarousel.css">
  <meta charset="utf-8" />

  <script src="CanvasInput.min.js"></script>
  <script src="p5.clickable.js"></script>
</head>

<body>

  <script src="sketch.js"></script>
  <script src="serial.js"></script>

</body>

</html>
```

The html script essentially just imports the different cdn libraries required to make the webpage run. The first library allows p5.js to get access to the webcam. The next one lets p5.js classify images and then the third one gives the webpage servo motor control. The next two libraries import the font that I used and then ‘CanvasInput.min.js’ and ‘p5.clickable.js‘ are the scripts for the webpage to have html5 access and the clickable buttons. Sketch.js is the script before that tells the browser what the webpage looks like, and serial.js gives the webpage serial access to talk to the Arduino.

P5.clickable.js

I didn't write the code for this script, it is a JavaScript library created by the developers of p5.js to add buttons to the webpage.

```
//Determines if the mouse was pressed on the previous frame
var cl_mouseWasPressed = false;
//Last hovered button
var cl_lastHovered = null;
//Last pressed button
var cl_lastClicked = null;
//All created buttons
var cl_clickables = [];

//This function is what makes the magic happen and should be ran after
//each draw cycle.
p5.prototype.runGUI = function(){
    for(i = 0; i < cl_clickables.length; ++i){
        if(cl_lastHovered != cl_clickables[i])
            cl_clickables[i].onOutside();
    }
    if(cl_lastHovered != null){
        if(cl_lastClicked != cl_lastHovered){
            cl_lastHovered.onHover();
        }
    }
    if(!cl_mouseWasPressed && cl_lastClicked != null){
        cl_lastClicked.onPress();
    }
    if(cl_mouseWasPressed && !mouseIsPressed && cl_lastClicked != null){
        if(cl_lastClicked == cl_lastHovered){
            cl_lastClicked.onRelease();
        }
        cl_lastClicked = null;
    }
    cl_lastHovered = null;
    cl_mouseWasPressed = mouseIsPressed;
}

p5.prototype.registerMethod('post', p5.prototype.runGUI);

//Button Class
function Clickable(){
    this.x = 0;         //X position of the clickable
    this.y = 0;         //Y position of the clickable
    this.width = 100;       //Width of the clickable
    this.height = 50;       //Height of the clickable
    this.color = "#FFFFFF";     //Background color of the clickable
    this.cornerRadius = 10;     //Corner radius of the clickable
    this.strokeWeight = 2;      //Stroke width of the clickable
    this.stroke = "#000000";    //Border color of the clickable
    this.text = "Press Me";     //Text of the clickable
    this.textColor = "#000000"; //Color for the text shown
    this.textSize = 12;     //Size for the text shown
    this.textFont = "sans-serif";   //Font for the text shown

    this.onHover = function(){
        //This function is ran when the clickable is hovered but not
        //pressed.
    }

    this.onOutside = function(){
        //This function is ran when the clickable is NOT hovered.
    }

    this.onPress = function(){
        //This fucking is ran when the clickable is pressed.
    }

    this.onRelease = function(){
        //This funcion is ran when the cursor was pressed and then
        //released inside the clickable. If it was pressed inside and
        //then released outside this won't work.
    }

    this.locate = function(x, y){
        this.x = x;
        this.y = y;
    }

    this.resize = function(w, h){
        this.width = w;
        this.height = h;
    }

    this.draw = function(){
        fill(this.color);
        stroke(this.stroke);
        strokeWeight(this.strokeWeight);

        rect(this.x, this.y, this.width, this.height, this.cornerRadius);
        fill(this.textColor);
        noStroke();
        textAlign(CENTER, CENTER);
        textSize(this.textSize);
        textFont(this.textFont);
        text(this.text, this.x+1, this.y+1, this.width, this.height);
        if(mouseX >= this.x && mouseX >= this.y
           && mouseX < this.x+this.width && mouseX < this.y+this.height){
            cl_lastHovered = this;
            if(mouseIsPressed && !cl_mouseWasPressed)
                cl_lastClicked = this;
        }
    }

    cl_clickables.push(this);
}
```

serial.js

```
(function() {
    'use strict';

    document.addEventListener('DOMNodeInserted', event => {
      let connectButton = document.querySelector("#connect");



      function connect() {
        port.connect().then(() => {

          connectButton.textContent = 'DISCONNECT';

          port.onReceive = data => {
            let textDecoder = new TextDecoder();
            console.log(textDecoder.decode(data));
          }
          port.onReceiveError = error => {
            console.error(error);
          };
        }, error => {

        });
      }
      try {
      connectButton.addEventListener('click', function() {
        if (port) {
          port.disconnect();
          connectButton.textContent = 'CONNECT ARDUINO';

          port = null;
        } else {
          serial.requestPort().then(selectedPort => {
            port = selectedPort;
            connect();
          }).catch(error => {

          });
        }
      });

      }
      catch € {

        console.log("p5 sketch not loaded yet: ", e);
      }

  //     serial.getPorts().then(ports => {
  //       if (ports.length == 0) {

  //       } else {

  //         port = ports[0];
  //         connect();
  //       }
  //     });
    });
  })();


  // From https://github.com/webusb/arduino/blob/gh-pages/demos/serial.js
  var serial = {};

  (function() {
    'use strict';

    serial.getPorts = function() {
      return navigator.usb.getDevices().then(devices => {
        console.log(devices);
        return devices.map(device => new serial.Port(device));
      });
    };

    serial.requestPort = function() {
      const filters = [
        { 'vendorId': 0x2341, 'productId': 0x8036 },
        { 'vendorId': 0x2341, 'productId': 0x8037 },
        { 'vendorId': 0x2341, 'productId': 0x804d },
        { 'vendorId': 0x2341, 'productId': 0x804e },
        { 'vendorId': 0x2341, 'productId': 0x804f },
        { 'vendorId': 0x2341, 'productId': 0x8050 },
      ];
      return navigator.usb.requestDevice({ 'filters': filters }).then(
        device => new serial.Port(device)
```

This script basically gives the Arduino Leonardo a serial connection to the p5.js sketch. The first function tries to pair the two by finding which port the Arduino is plugged into and requesting to connect.

Once the Arduino is connected the 'CONNECT ARDUINO' button will change to say 'DISCONNECT'. The catch(e) statement allows you to handle the error of 'p5 sketch not loaded yet' in the Google Chrome console if you aren't running the sketch in editing mode. This means that you won't need to have two tabs open at once while running the serial connection.

I didn't write the rest of this code below, I got it from github, but what it does is it asks the Arduino to connect or disconnect.

```
    );
  }

  serial.Port = function(device) {
    this.device_ = device;
  };

  serial.Port.prototype.connect = function() {
    let readLoop = () => {
      this.device_.transferIn(5, 64).then(result => {
        this.onReceive(result.data);
        readLoop();
      }, error => {
        this.onReceiveError(error);
      });
    };

    return this.device_.open()
        .then(() => {
          if (this.device_.configuration === null) {
            return this.device_.selectConfiguration(1);
          }
        })
        .then(() => this.device_.claimInterface(2))
        .then(() => this.device_.selectAlternateInterface(2, 0))
        .then(() => this.device_.controlTransferOut({
            'requestType': 'class',
            'recipient': 'interface',
            'request': 0x22,
            'value': 0x01,
            'index': 0x02}))
        .then(() => {
          readLoop();
        });
  };

  serial.Port.prototype.disconnect = function() {
    return this.device_.controlTransferOut({
            'requestType': 'class',
            'recipient': 'interface',
            'request': 0x22,
            'value': 0x00,
            'index': 0x02})
        .then(() => this.device_.close());
  };

  serial.Port.prototype.send = function(data) {
    return this.device_.transferOut(4, data);
  };
})();
```


This Arduino sketch is used to connect the Arduino board to the webpage and move the servo motor when the AI detects something. It also prints what it's doing and what it has detected in the serial monitor when it classifies something.

```
#include <WebUSB.h>
#include <Servo.h>

//TODO: fix this url hinting
WebUSB WebUSBSerial(1 /* https:// */, "webusb.github.io/arduino/demos/rgb");

#define Serial WebUSBSerial
Servo myservo;

const int redPin = 9;
const int greenPin = 10;
const int bluePin = 11;
int pos = 0;    // variable to store the servo position

int color[3];
int colorIndex;

void setup() {
  while (!Serial) {
    ;
  }
  Serial.begin(9600);
  Serial.write("Sketch begins.\r\n");
  Serial.flush();
  colorIndex = 0;
  myservo.attach(9);
  myservo.write(60);
}

void loop() {

  if (Serial && Serial.available()) {
    color[colorIndex++] = Serial.read();
    if (colorIndex == 1) {

      //       Serial.flush();
      //      analogWrite(redPin, color[0]);
      if (color[0] == 1) {
        myservo.write(0);
        delay(2000);
        for (pos = 0; pos <= 75; pos += 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo.write(pos);              // tell servo to go to position in variable 'pos'
          delay(5);                       // waits 15ms for the servo to reach the position
        }
        delay(1000);
        Serial.write("ceral detected.\r\n");
      }

      else if (color[0] == 2) {
        myservo.write(180);
        delay(2000);
        for (pos = 180; pos >= 75; pos -= 1) { // goes from 0 degrees to 180 degrees
          // in steps of 1 degree
          myservo.write(pos);              // tell servo to go to position in variable 'pos'
          delay(20);                       // waits 15ms for the servo to reach the position
        }
        delay(1000);
        Serial.write("mallow detected.\r\n");
      }
      while (Serial.available()) {
        int throwaway = Serial.read();
        Serial.write("Throwing away.\r\n");
      }

      Serial.flush();
      colorIndex = 0;
    }
  }
  else {
    //    Serial.println("asdf");
    for (pos = 60; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees
      // in steps of 1 degree
      myservo.write(pos);              // tell servo to go to position in variable 'pos'
      delay(3);                       // waits 15ms for the servo to reach the position
    }
    for (pos = 90; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
      myservo.write(pos);              // tell servo to go to position in variable 'pos'
      delay(3);                       // waits 15ms for the servo to reach the position
    }
//    delay(200);
  }


}
```

The Arduino code starts off by importing the required libraries. WebUSB allows the Arduino Leonardo to connect to the webpage through USB connection, and Servo.h gives the board servo control.

Here it is setting up the serial monitor and prints 'Sketch begins.\r\n'. Then it sets the servo motor to 60°. The first loop statement states that if it detects class 1 it will turn from 75 degrees higher than what it currently is. Otherwise if it detects class 2 it will turn 75 degrees the other way. While the servo motor is turning it will print that it is throwing it away.

This part of the code states that if there is nothing there the servo motor will go back and forth between 60 and 90 degrees until it gets the signal that the AI detects something.

The next section of code creates the runlinc webpage that displays the firetruck's statistics. Runlinc is a web application created locally in SA by eLabtronics, a part of the STEMSEL club that I go to every Saturday. Essentially it is flashed onto an ESP8266 Wi-Fi chip, which can be plugged in to a STEMSEL board. Once connected to Wi-Fi a webpage linked to the chip can be opened from anywhere and used to either program something or run code.

CSS

```
html * {
	font-family: Arial;
	font-weight: bold;
}

html body {
	background: rgb(0,0,0);
background: linear-gradient(180deg, rgba(0,0,0,1) 0%, rgba(0,0,0,1) 19%, rgba(255, 188, 5) 22%, rgba(177,
89, 16) 84%, rgba(0,0,0,1) 87%, rgba(0,0,0,1) 100%);
}
```

In this first section I am declaring the font that I'm using, and that it's bold.

Now I'm declaring the colour of the body of the background. The three numbers in the brackets is the rgb of the colour that I am using. I am using size 12.

```
div pre {
	font-family: Arial;
	font-weight: normal;
	font-size: 12px;
}

.myButton {
	-moz-box-shadow: 0px 0px 0px 2px #d4762a;
	-webkit-box-shadow: 0px 0px 0px 2px #d4762a;
	box-shadow: 0px 0px 0px 2px #d4762a;
	background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #d4762a;), color-stop(1,
#d4762a));
	background:-moz-linear-gradient(top, #d4762a 5%, #b81111 100%);
	background:-webkit-linear-gradient(top, #d4762a 5%, #b81111 100%);
	background:-o-linear-gradient(top, #d4762a 5%, #b81111 100%);
	background:-ms-linear-gradient(top, #d4762a 5%, #b81111 100%);
	background:linear-gradient(to bottom, #d4762a 5%, #b81111 100%);
	filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#d4762a',
endColorstr='#b81111',GradientType=0);
	background-color:#d4762a;
	-moz-border-radius:10px;
	-webkit-border-radius:10px;
	border-radius:10px;
	border:1px solid #d4762a;
	display:inline-block;
	cursor:pointer;
	color: #d4762a;
	font-family: 'Electrolize', sans-serif;
	font-weight: bold;
	font-size:15px;
	padding:8px 15px;
	margin: auto;
	margin-top: 20px;
	display: flex; justify-content: center;
	text-decoration:none;
	text-shadow:0px 1px 0px #aade7c;
}
```

This section of the CSS is the colour of my reset button. It also includes the border, the gradients, the. Font and the size.

```
.myButton:hover {
	background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #d4762a), color-stop(1,
#d4762a));
	background:-moz-linear-gradient(top, #d4762a 5%, #d4762a 100%);
	background:-webkit-linear-gradient(top, #d4762a 5%, #d4762a 100%);
	background:-o-linear-gradient(top, #d4762a; 5%, #d4762a 100%);
	background:-ms-linear-gradient(top, #d4762a; 5%, #d4762a 100%);
	background:linear-gradient(to bottom, #d4762a; 5%, #d4762a 100%);
	filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#d4762a;
endColorstr='#d4762a;',GradientType=0);
	background-color:#d4762a;
.myButton:active {
	position:relative;
	top:1px;
}

.group {
-webkit-box-shadow:0px 0px 5px 3px #d4762a;
-moz-box-shadow:0px 0px 5px 3px #d4762a;
box-shadow:0px 0px 5px 3px #d4762a;
	background: rgb(212, 118, 42); /* Old browsers */
background: -moz-linear-gradient(top, rgba(212, 118, 42) 0%, rgba(212, 118, 42) 17%, rgba(212, 118, 42)
90%);
background: -webkit-linear-gradient(top, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118, 42
90%);
background: linear-gradient(to bottom, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118, 42)
90%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#d4762a',
endColorstr='#d4762a',GradientType=0 );
}

.col-header {
	background: #d4762a;
background: -moz-linear-gradient(top, #d4762a 0%, #d4762a 50%, #d4762a 96%, #d4762a 100%);
background: -webkit-linear-gradient(top, #d4762a 0%,#d4762a 50%,#d4762a 96%,#d4762a 100%);
background: linear-gradient(to bottom, #d4762a 0%,#d4762a 50%,#d4762a 96%,#d4762a 100%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#d4762a',
endColorstr='#d4762a',GradientType=0 );
	height: 25px;
	padding-top: 5px;
}

.col-row {
	height: 340px;
	background: rgb(212, 118, 42);
background: -moz-linear-gradient(top, rgba(212, 118, 42) 0%, rgba(212, 118, 42) 17%, rgba(212, 118, 42)
90%);
background: -webkit-linear-gradient(top, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118, 42)
90%);
background: linear-gradient(to bottom, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118, 42)
90%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#d4762a',
endColorstr='#d4762a',GradientType=0 );
	border: 1px solid black;
	margin: 0.5px;
}

.center {
 margin:auto;
}
```

This whole page is getting the colour of the background right.

```
.rounded {
border-radius:10px;
-moz-border-radius:10px;
-webkit-border-radius:10px;
}

.rounded-top-left {
border-radius:10px 0 0 10px;
-moz-border-radius:10px 0 0 10px;
-webkit-border-radius:10px 0 0 10px;
}

.rounded-top-right {
border-radius:10px 10px 0 0;
-moz-border-radius:10px 10px 0 0;
-webkit-border-radius:10px 10px 0 0;
}

.rounded-bottom-left {
border-radius: 0 0 10px 10px;
-moz-border-radius: 0 0 10px 10px;
-webkit-border-radius: 0 0 10px 10px;
}

.rounded-bottom-right {
border-radius: 0 10px 10px 0;
-moz-border-radius: 0 10px 10px 0;
-webkit-border-radius: 0 10px 10px 0;
}

.dropdown {
	background: rgb(255, 208, 0);
	background: -moz-linear-gradient(top, rgba(212, 118, 42) 0%, rgba(212, 118, 42) 17%, rgba(212, 118,
42) 90%);
	background: -webkit-linear-gradient(top, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118,
42) 90%);
	background: linear-gradient(to bottom, rgba(212, 118, 42) 0%,rgba(212, 118, 42) 17%,rgba(212, 118, 42)
90%);
	filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#d4762a',
endColorstr='#d4762a',GradientType=0 );
}

select {
  -webkit-padding-end: 20px;
  -moz-padding-end: 20px;
  -webkit-padding-start: 2px;
  -moz-padding-start: 2px;
  background-position: center right;
  border: 1px solid #AAA;
  border-radius: 2px;
  box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.1);
  color: #555;
  font-size: inherit;
  margin: 0;
  overflow: hidden;
  padding-top: 2px;
  padding-bottom: 2px;
  text-overflow: ellipsis;
  white-space: nowrap;
}
```

Here I am declaring the colour scheme of the table.

PRINCE ALFRED COLLEGE 2020
CALEB TANG

HTML

```
<div class="center body" style="text-align:center; width:510px;">
  <h2>
    <font color="orange">Smart Recycling</font>
  </h2>
</div>
<div class="group center rounded" style="text-align:left; background-color:#ffcc00; width:250px;
padding:10px;">
  <td>Current Temperature:</td>
  <td>
    <font id="Temp"></font> °C
  </td>
  <br>
  <td>Waste Thrown:</td>
  <td>
    <font id="Pcurrent"></font>
  </td>
  <br>
  <td>Fertiliser Weight:</td>
  <td>
    <font id="Wcurrent"></font> Kg
  </td>
  <br>
</div>
<div style="text-align:center">
  <br>
  <div class="center" style="display:inline-block; margin-top: 10px;">
    <td><label>Interval</label></td>
    <td>
      <select class="dropdown" name="interval" onchange="
          interval = value * 1000; clearTimeout(timer);
          tempLogger();
        " id="setInterval" alt="Interval">
        <option value="1" selected>1 Second</option>
        <option value="10">10 Seconds</option>
        <option value="60">1 Minute</option>
        <option value="600">10 Minutes</option>
      </select>
    </td>
  </div>
  <div class="center" style="text-align:center; width:570px;">
    <h3>Data Logging Table</h3>
  </div>
  <div class="center" style="display:flex; width: 570px;">
    <div class = "group col-row rounded-top-left" id="time" style="overflow:hidden; width:150px;"
align="center">
      <div class = "col-header" style="width:150px;">
        Time
      </div>
      <pre id="Time" align="center">
        </pre>
    </div>
    <div class = "group col-row" align="center" id="temp" style="width:150px;">
      <div class = "col-header" style="width:150px;">
        Temperature
      </div>
      <pre id="tempPre" align="center">
        </pre>
    </div>
    <div class = "group col-row" align="center" id="weight" style="width:120px;">
      <div class = "col-header" style="width:120px;">
        Weight
      </div>
      <pre id="weightP" align="center">
        </pre>
    </div>
  </div>
</div>
<div style="text-align:center">
 <button class="myButton" onclick="clearTable()" style="background-color: #4CAF50;">Reset</button>
</div>
```

Here I am declaring the HTML of my webpage. After making my title orange, I get the table of readings done. The first section is the announcements box, but I disabled it, as seen as in the loop. Then I get my temperature logging and time working.

This is where I get the option to choose the interval at which the table's readings are shown.

Here is where I put the results by using "<pre id = >"

This section is where I make it so that when I press the reset button it resets the table.

CALEB TANG

JavaScript Select Macro | select a device | Add Macro

```
var resistor = 160;
var interval = 1000;
var profit = 0;
var log = 0;
var pop = 0;
var loggerT = "";
var loggerW = "";
var loggerPP = "";
var loggerTT = "";
var timer = setTimeout("tempLogger()", interval);
var counter = 0;


function getTemp() {
  temp = analogIn(tempSensor);
  temp = Math.log((255 / temp - 1) * resistor / 316.25) / 0.045;
  degree = temp.toFixed(1);
  return degree;
}

function getDate() {
  now = new Date();
  year = now.getFullYear();
  month = now.getMonth() + 1;
  day = now.getDate();
  hour = now.getHours();
  minute = now.getMinutes();
  second = now.getSeconds();

  d = (
    year + "/" +
    month + "/" +
    day + " " +
    hour + ":" +
    minute + ":" +
    second
  );
  return d;
}

var msg = new SpeechSynthesisUtterance("Warning, Rubbish is tipping");
var voices = window.speechSynthesis.getVoices();
msg.voice = voices[0];

function tempLogger() {
  T = document.getElementById('tempPre');
  W = document.getElementById('weightP');
  tt = document.getElementById('Time');
  if (log < 20) {
    log++;
    loggerT = loggerT + getTemp() + " °C" + "\n";
    loggerW = loggerW + analogIn(weight) + " Kg" + "\n";
    loggerTT = loggerTT + getDate() + "\n";

    T.innerHTML = loggerT;
    W.innerHTML =  loggerW;
    tt.innerHTML = loggerTT;
  } else {
    resetTable();
  }
  timer = setTimeout("tempLogger()", interval);
}

function resetTable() {
  T = document.getElementById('tempPre');
  W = document.getElementById('weightP');
  pp = document.getElementById('popu');
  tt = document.getElementById('Time');
  loggerT = "";
  loggerW = "";
  loggerPP = "";
  loggerTT = "";
  log = 0;
  T.innerHTML = "";
  W.innerHTML = "";
  pp.innerHTML = "";
  tt.innerHTML = "";
}

function clearTable() {
  resetTable();
  pop = 0;
}
```

This is where I declare my variables. Var resistor is for the resistance that the temperature sensor is at. This can be changed to alter the temperature. Var log is to log the table.

The function ”getTemp()” is the formula that I use to get the temperature. The function “getDate()” is the formula that I use to get the current time.

The variable “msg” is the variable that I use to get the computer to give a warning.

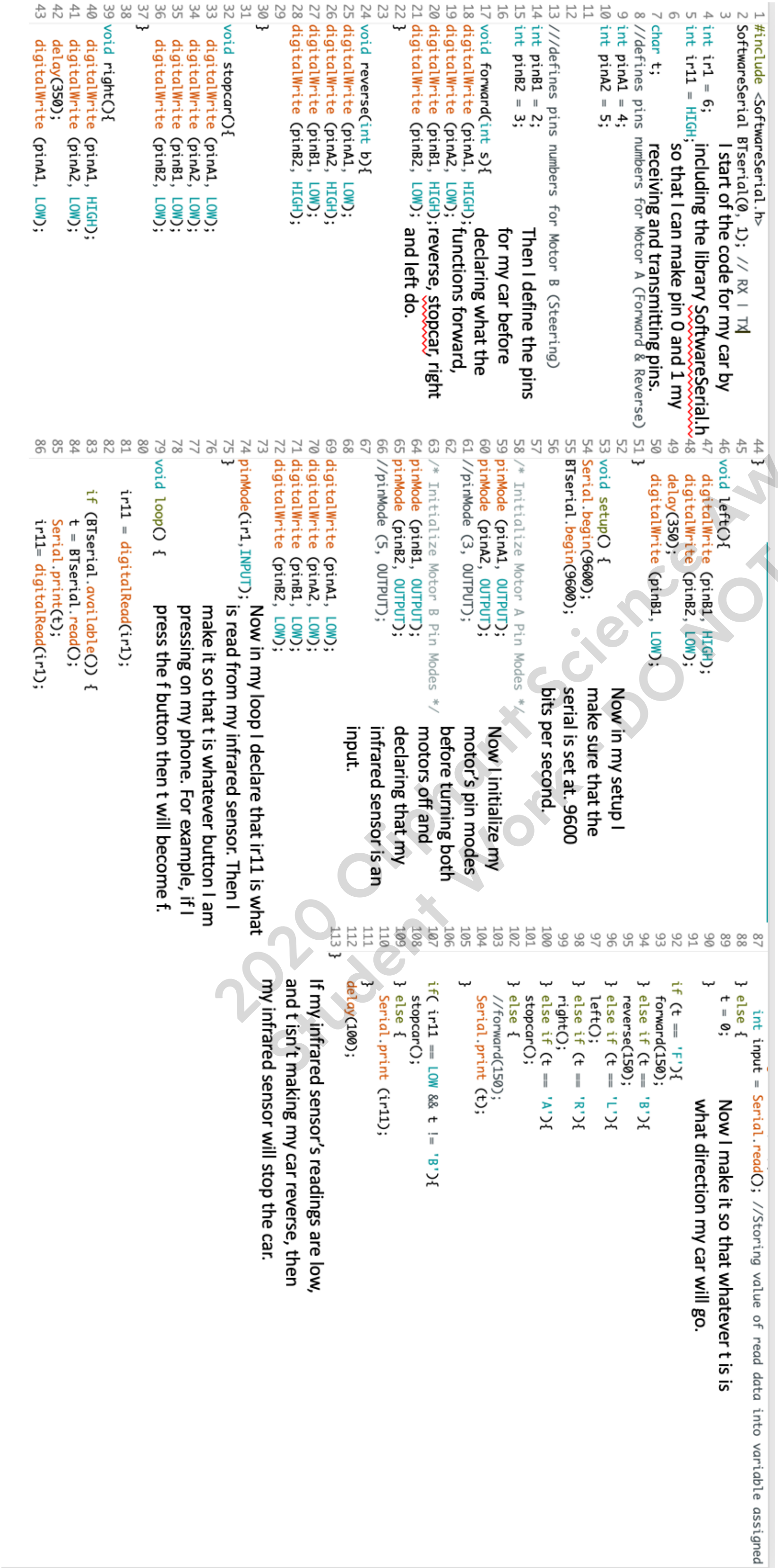
I use the function “resetTable() to clear the table when the reset button is pressed.

JavaScript Loop Select Macro | select a device | Add Macro

```
//document.getElementById('Temp').innerHTML = getTemp();

theWeight = analogIn(weight);

//document.getElementById('Wcurrent').innerHTML = theWeight;
//document.getElementById('fertiliser').innerHTML = profit;

if (analogIn(LightSensor1) <20) {
  turnOn(greenLed);
} else {
  turnOff(greenLed);
}

if (analogIn(LightSensor2) <20) {
  turnOn(yellowLed);
} else {
  turnOff(yellowLed);
}

if (analogIn(LightSensor3) <20) {
  turnOn(redLed);
} else {
  turnOff(redLed);
}

if(digitalIn(TiltSensor) == 1) {
 window.speechSynthesis.speak(msg);
} else{
 window.speechSynthesis.cancel();
}
```

The // bits are the parts for the notices board. The analogIn declares the potentiometer’s name as ‘weight’. Then my code goes into a series of decisions.

This section of Arduino code is used to actually drive the truck. Using Bluetooth, the Arduino is able to be connected to a mobile app where the driver has fast, responsive control of the vehicle.



```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(0, 1); // RX | TX

int ir1 = 6;
int ir11 = HIGH;

char t;
//defines pins numbers for Motor A (Forward & Reverse)
int pinA1 = 4;
int pinA2 = 5;


///defines pins numbers for Motor B (Steering)
int pinB1 = 2;
int pinB2 = 3;

void forward(int s){
digitalWrite (pinA1, HIGH);
digitalWrite (pinA2, LOW);
digitalWrite (pinB1, HIGH);
digitalWrite (pinB2, LOW);
}

void reverse(int b){
digitalWrite (pinA1, LOW);
digitalWrite (pinA2, HIGH);
digitalWrite (pinB1, LOW);
digitalWrite (pinB2, HIGH);

}

void stopcar(){
  digitalWrite (pinA1, LOW);
  digitalWrite (pinA2, LOW);
  digitalWrite (pinB1, LOW);
  digitalWrite (pinB2, LOW);
}

void right(){
  digitalWrite (pinA1, HIGH);
  digitalWrite (pinA2, LOW);
  delay(350);
  digitalWrite (pinA1, LOW);
}

void left(){
  digitalWrite (pinB1, HIGH);
  digitalWrite (pinB2, LOW);
  delay(350);
  digitalWrite (pinB1, LOW);
}

void setup() {
Serial.begin(9600);
BTserial.begin(9600);


/* Initialize Motor A Pin Modes */
pinMode (pinA1, OUTPUT);
pinMode (pinA2, OUTPUT);
//pinMode (3, OUTPUT);

/* Initialize Motor B Pin Modes */
pinMode (pinB1, OUTPUT);
pinMode (pinB2, OUTPUT);
//pinMode (5, OUTPUT);


digitalWrite (pinA1, LOW);
digitalWrite (pinA2, LOW);
digitalWrite (pinB1, LOW);
digitalWrite (pinB2, LOW);

pinMode(ir1,INPUT);
}



void loop() {

  ir11 = digitalRead(ir1);

  if (BTserial.available()) {
      t = BTserial.read();
      Serial.print(t);
      ir11= digitalRead(ir1);
    int input = Serial.read(); //Storing value of read data into variable assigned
  } else {
    t = 0;
  }

  if (t == 'F'){
    forward(150);
  } else if (t == 'B'){
    reverse(150);
  } else if (t == 'L'){
    left();
  } else if (t == 'R'){
    right();
  } else if (t == 'A'){
    stopcar();
  } else {
    //forward(150);
    Serial.print (t);
  }

  if( ir11 == LOW && t != 'B'){
    stopcar();
  } else {
    Serial.print (ir11);
  }
  delay(100);
}
```

I start of the code for my car by including the library SoftwareSerial.h so that I can make pin 0 and 1 my receiving and transmitting pins.

Then I define the pins for my car before declaring what the functions forward, reverse, stopcar, right and left do.

Now in my setup I make sure that the serial is set at. 9600 bits per second.

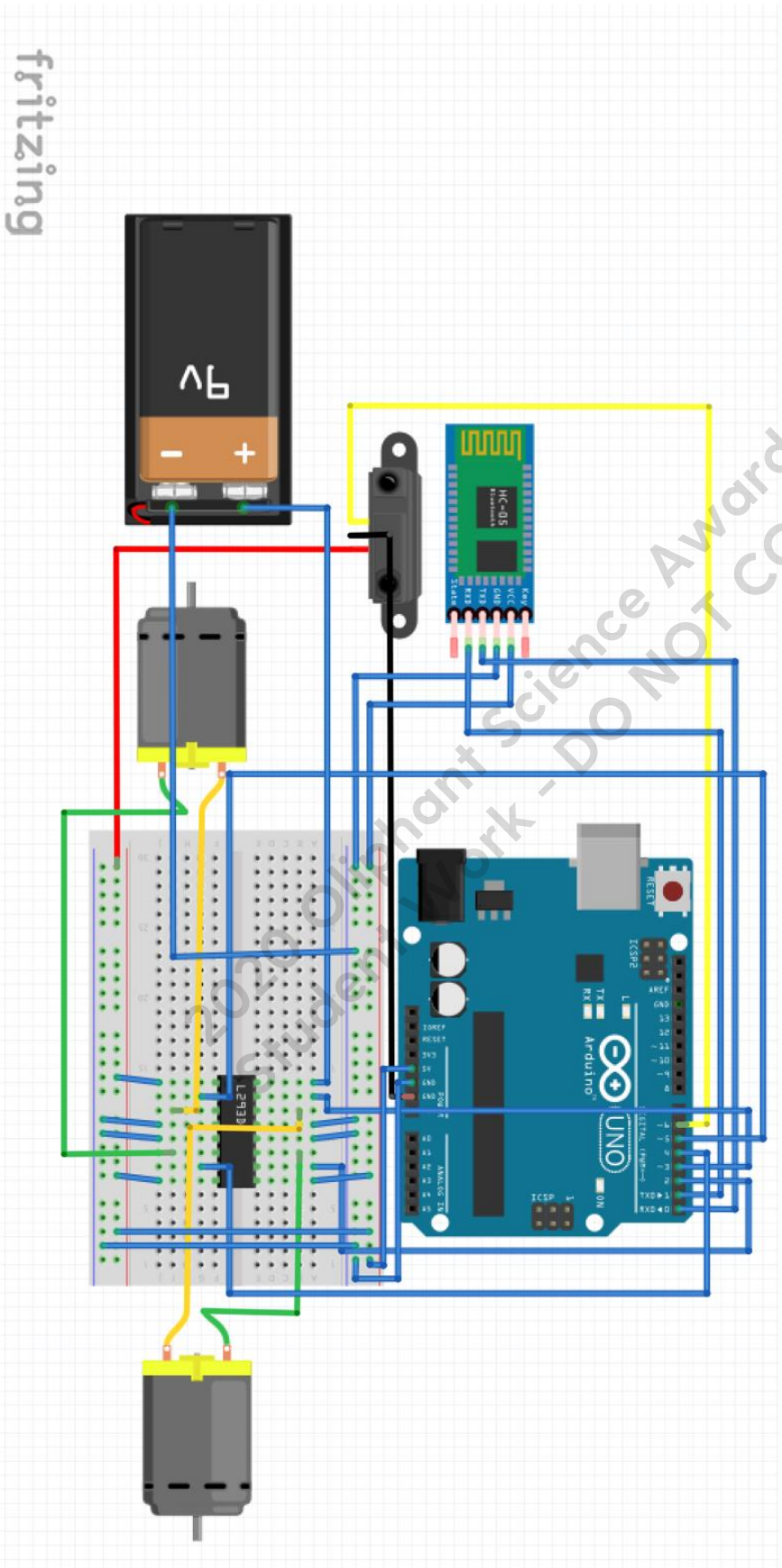
Now I initialize my motor's pin modes before turning both motors off and declaring that my infrared sensor is an input.

Now in my loop I declare that ir11 is what is read from my infrared sensor. Then I make it so that t is whatever button I am pressing on my phone. For example, if I press the f button then t will become f.

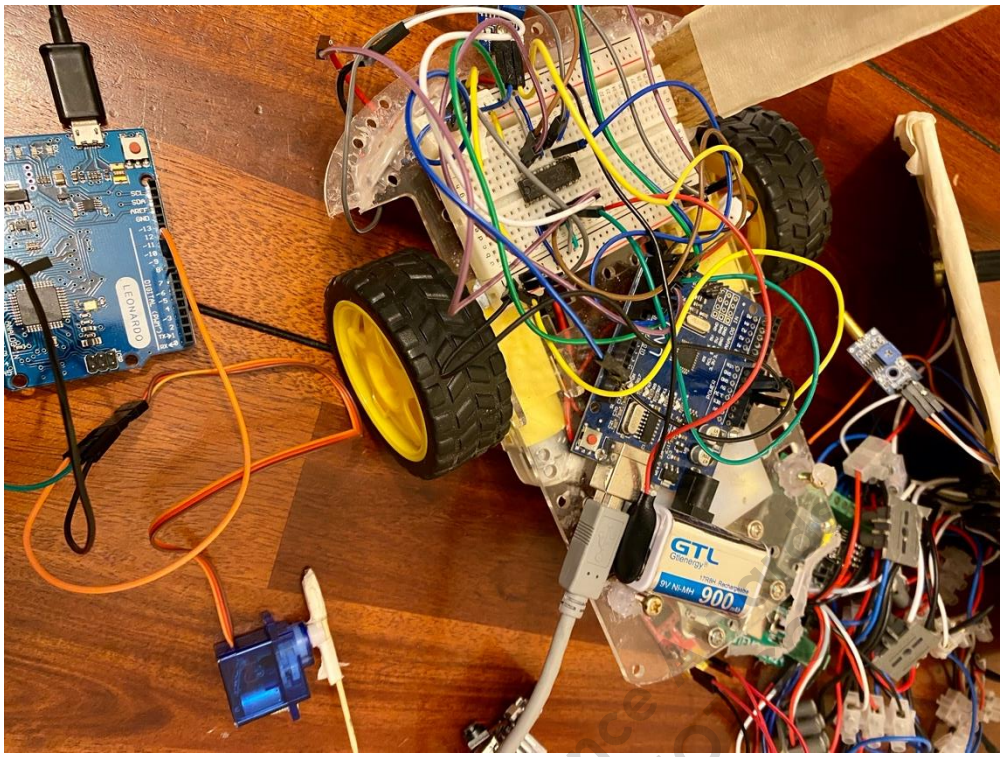
Now I make it so that whatever t is is what direction my car will go.

If my infrared sensor's readings are low, and t isn't making my car reverse, then my infrared sensor will stop the car.

This schematic is a rough idea of what the wiring of the truck should look like.



This is a picture of what my incomplete project looks like so far:



Acknowledgements

I am extremely grateful for the help that I have received from the several online forums that I have used, Arduino, GitHub and Stack Overflow in particular. Medium and Towards Data Science articles have also been highly useful. I would also like to acknowledge STEMSEL for helping me debug my code, and of course my parents for buying me parts that I need.

References

En.wikipedia.org. (2019). *Inductive charging*. [online] Available at: https://en.wikipedia.org/wiki/Inductive_charging [Accessed 12 Jul. 2019].

Forum.arduino.cc. (2019). *Arduino Forum - Index*. [online] Available at: https://forum.arduino.cc/.

Question, C., ..., F., t..., C., Production..., H. and promp..., D. (2019). *GitHub Community Forum*. [online] Github.community. Available at: https://github.community/.

US EPA. (2019). *What if Garbage Fumes Powered More of Our Cars, Trucks, and Buses? | US EPA*. [online] Available at: https://www.epa.gov/greenvehicles/what-if-garbage-fumes-powered-more-our-cars-trucks-and-buses [Accessed 12 Jul. 2019].

Aph.gov.au. 2020. *2019–20 Australian Bushfires—Frequently Asked Questions: A Quick Guide – Parliament Of Australia*. [online] Available at: <https://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/rp/rp1920/Quick_Guides/AustralianBushfires>. [Accessed 12 Jul. 2019]

Duncombe, J., 2020. *Five Environmental Consequences Of Australia'S Fires - Eos*. [online] Eos. Available at: <https://eos.org/articles/five-environmental-consequences-of-australias-fires> [Accessed 18 July 2020].