Pest Alert

Group 22
Aisha Ben-Neticha (EE)
Owen Boucher (EE)
Angela Wong (CompE)
Madelyn Wright (EE)

Problem Statement

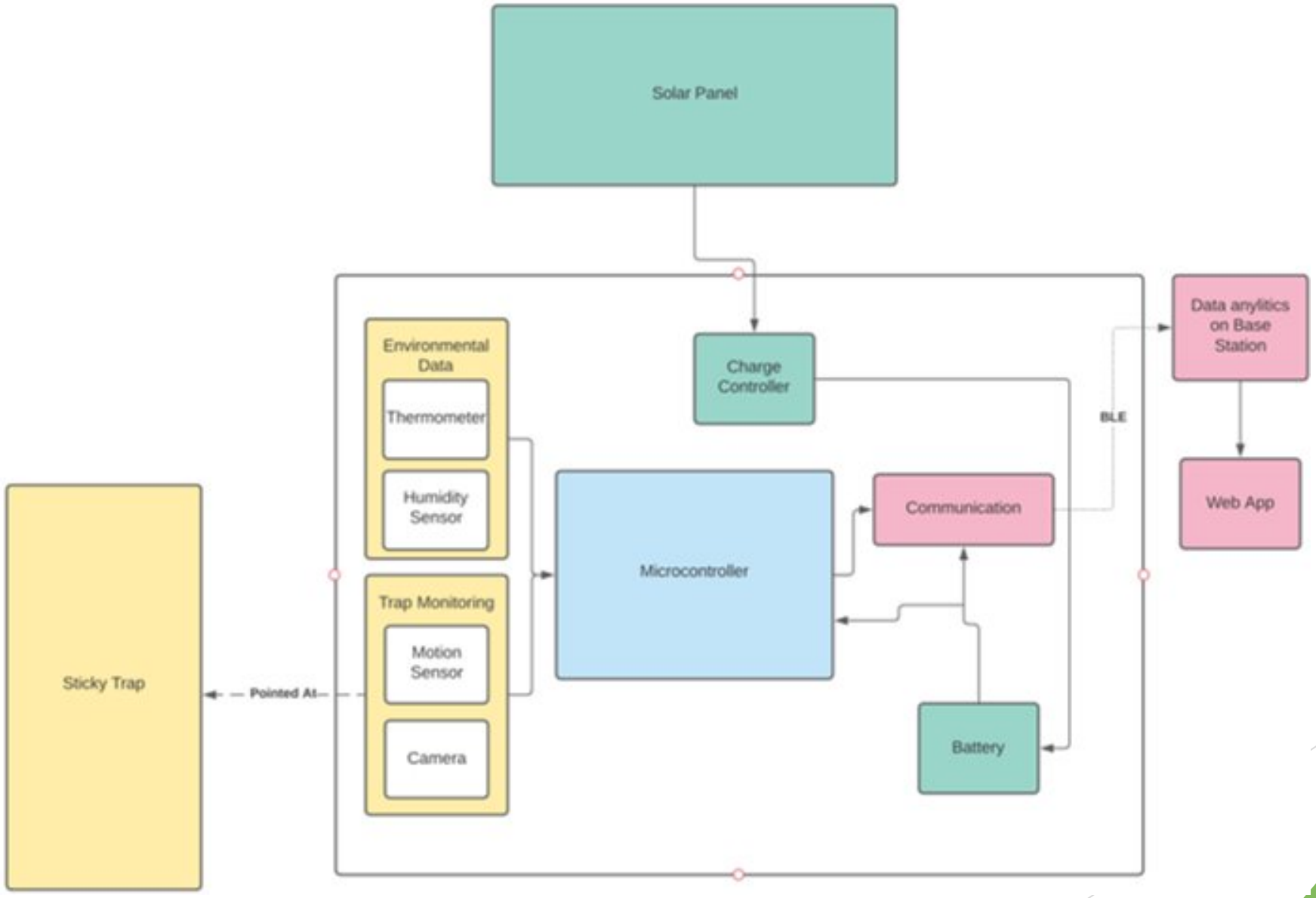
According to the United Nations Food and Agriculture Organization, between 20 and 40 percent of food crops are lost each year to pests. Integrated Pest Management, or IPM, can help reduce this loss. One of the major problems facing IPM is that it requires a huge labor force to monitor pests.

Our system, Pest Alert, seeks to create an **automated**, **low cost**, **low power**, and **low maintenance** system for monitoring pests in comparison to its competitors. Our system will automatically notify farmers about possible infestations and the environmental states that led to it, by using traps, images, and sensors, and sending that information to a base station. On top of this, the system will be **managing its own power usage** with a solar panel and battery.

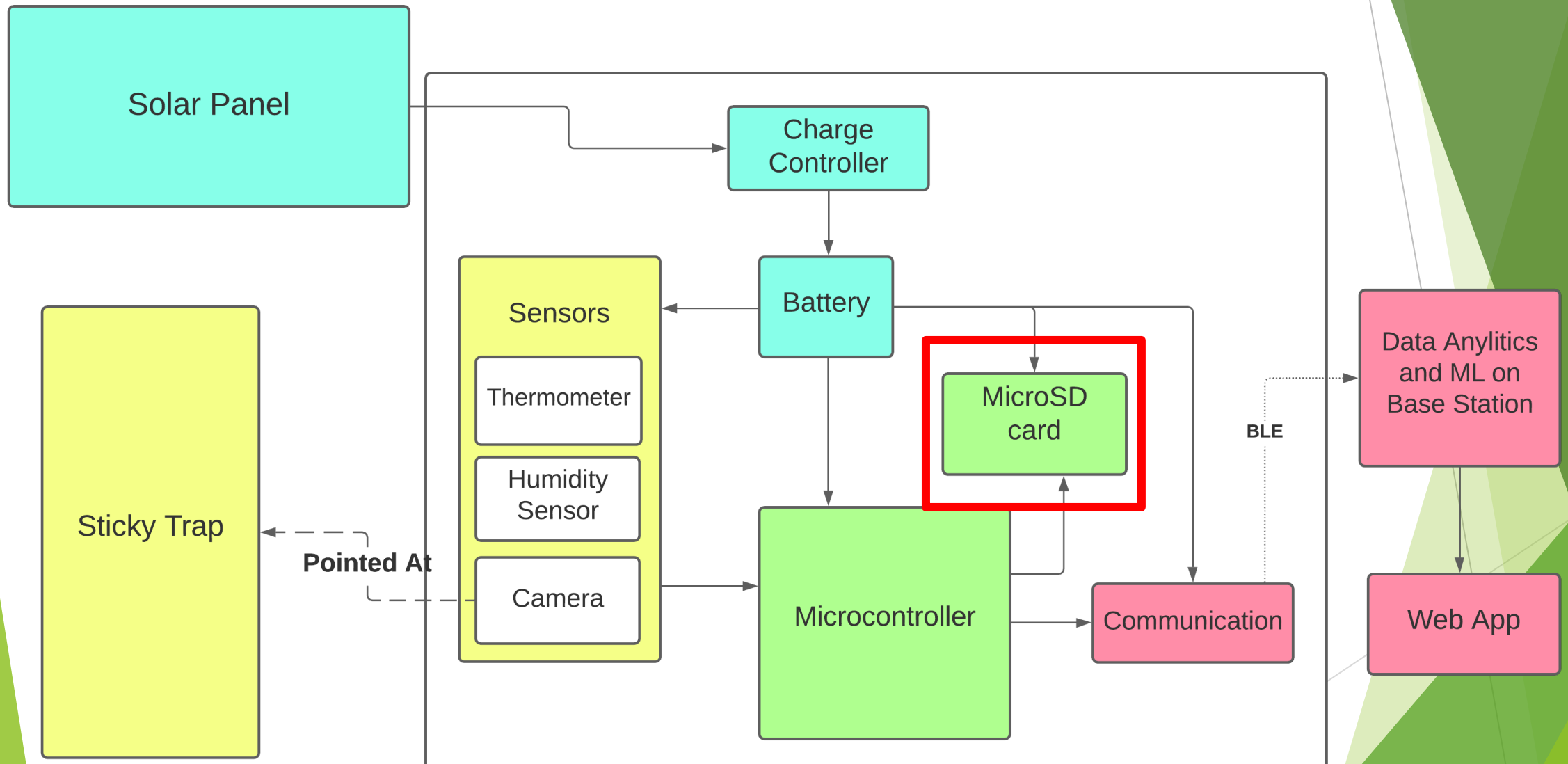
System Specifications

- System must be operable in the range of 0ºC to 50ºC
- Sensors
  - Low powered camera to capture images of and identify pests on a sticky trap (<5mW)
  - Temperature/Humidity sensors to collect surrounding environmental data and observe correlation between environment and pest accumulation
- Power
  - Battery life of 24 hours on full charge: Used to power sensors and should last at least two months (with charging) without any interference
  - Solar Panel: Recharge half of the battery with one day of sunlight
  - Reduce monitoring at night and operate in very low power mode by reducing overall operation because energy cannot be harvested in the dark
- Traps should be replaced every two months, when they are full, or when they lose their stickiness.
- Low power communication between sensors and server using Bluetooth Low Energy (<100mW)
- Machine learning/analysis will be able to run on the base station

Old Hardware Design

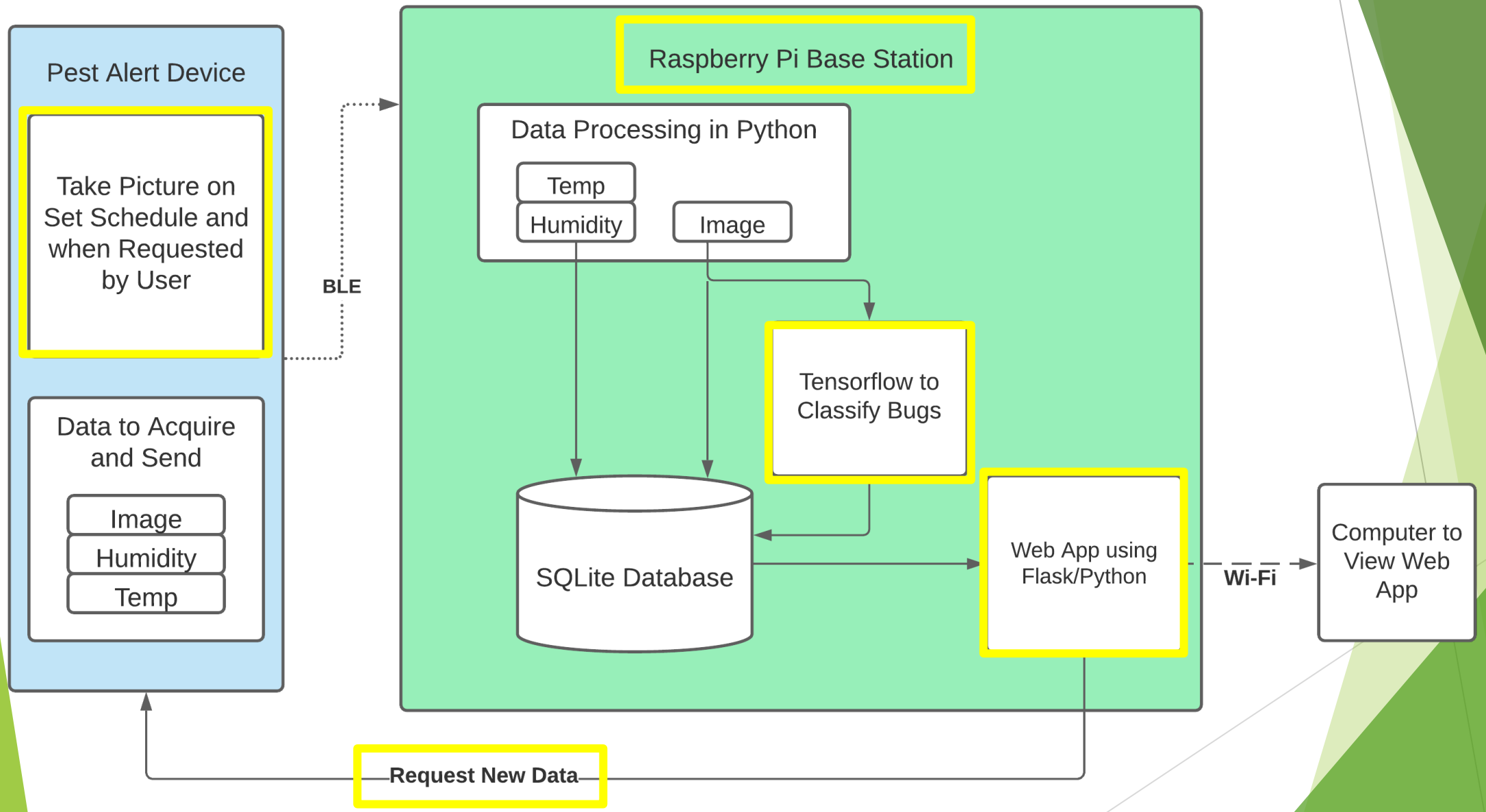


Hardware Design



** Motion sensor has been removed, MicroSD card has been added (from previous design) **

Software Design



List of Hardware and Software

Current

- Adafruit MicroSD Breakout Board
- Adafruit Bluefruit BLE UART Friend
- TC4056A Battery Charging Module
- 18650 Cell Battery
- 1W Solar Panel
- TTL Serial Camera Module
- DHT11 Temperature and Humidity Sensor Module
- Raspberry Pi 3B+
- Arduino Due
- Tensorflow
- Flask/Python
- SQLite

Old MDR Deliverables

Owen:

- BLE Communication established
- Framework of visual representation

Aisha:

- Temperature, humidity, and motion detecting sensors configured and functioning (w/ 70% accuracy)
- Camera is configured; can capture images when motion sensor is triggered

Angela:

- Database configured, data conditioning is successful
- 70% accuracy in bug detection

Madelyn:

- Connection between solar panel and battery made
- Code configured to operate in low power modes
- Panel can successfully supply power to sensors

MDR Deliverables

Owen:

- ✓ Data conditioning
- ✓ **PRIORITY:** BLE Communication established

Aisha:

- ✓ Humidity and temperature sensors working
- ✓ **PRIORITY:** Camera is configured to capture images with the press of a button

Angela:

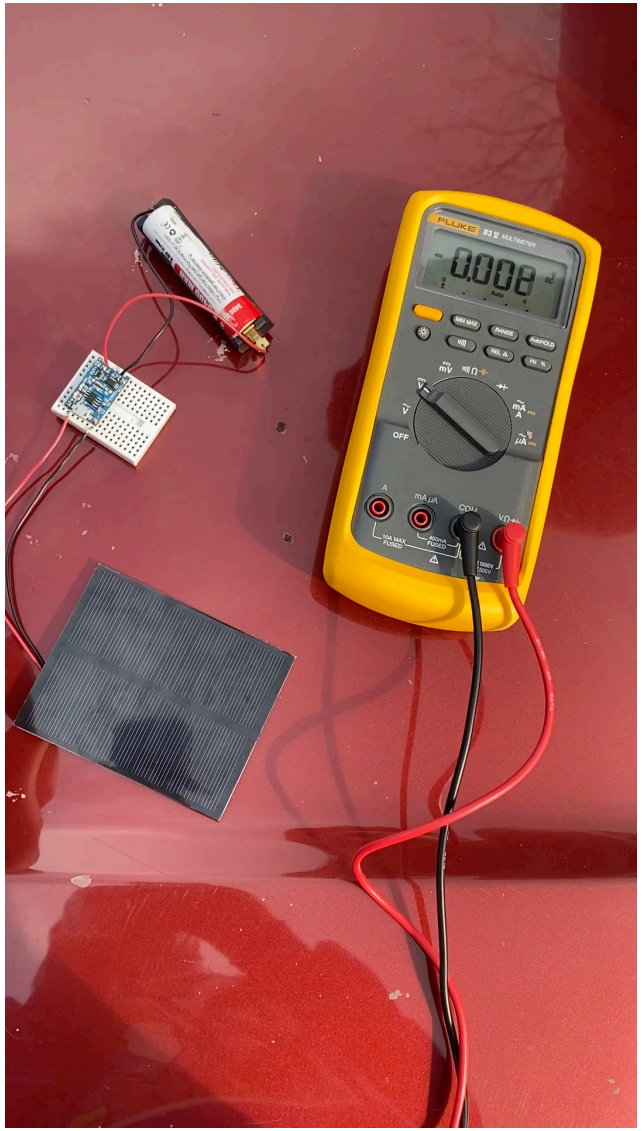
- ✓ Database configured
- ✓ **PRIORITY:** 70% accuracy in bug classification

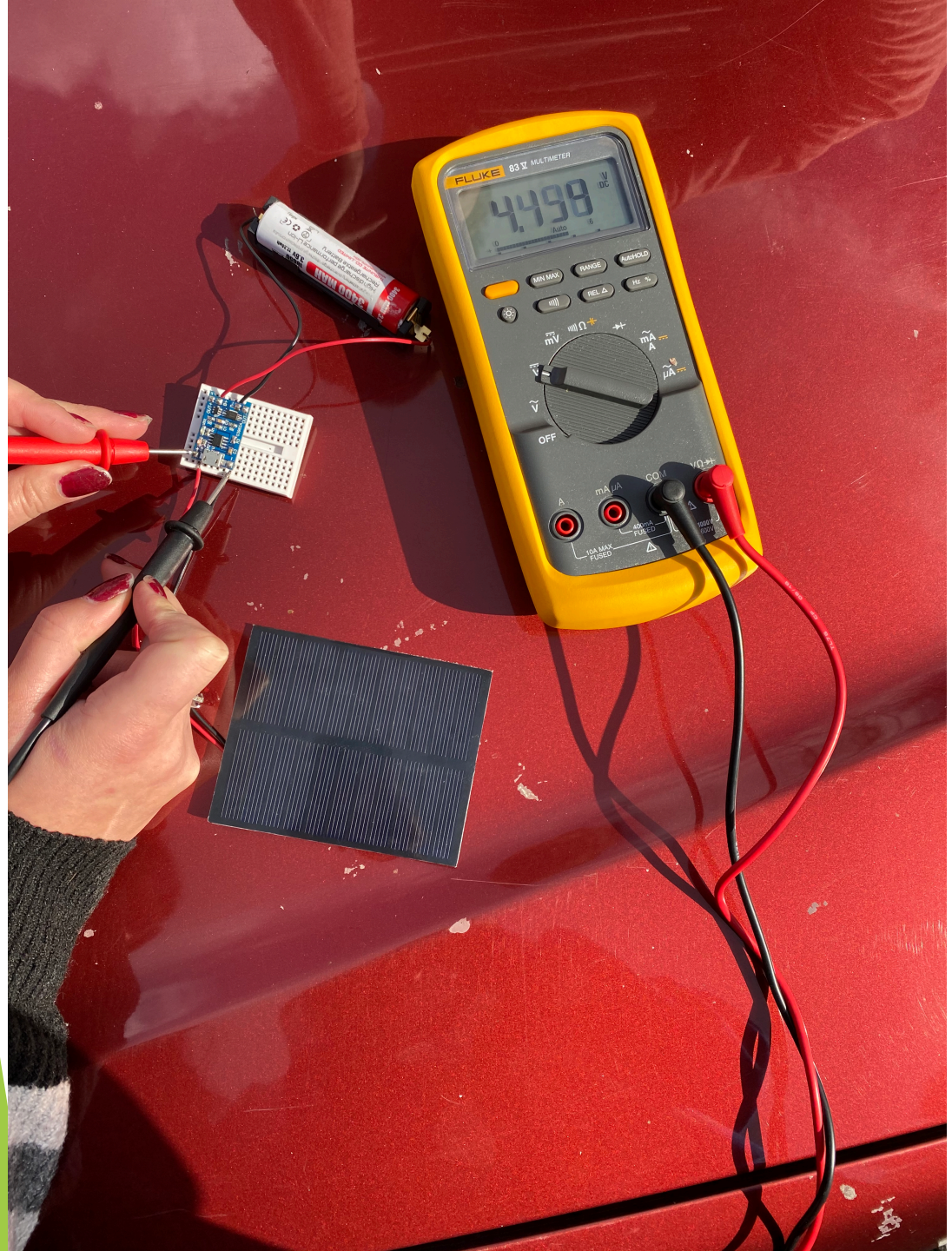
Madelyn:

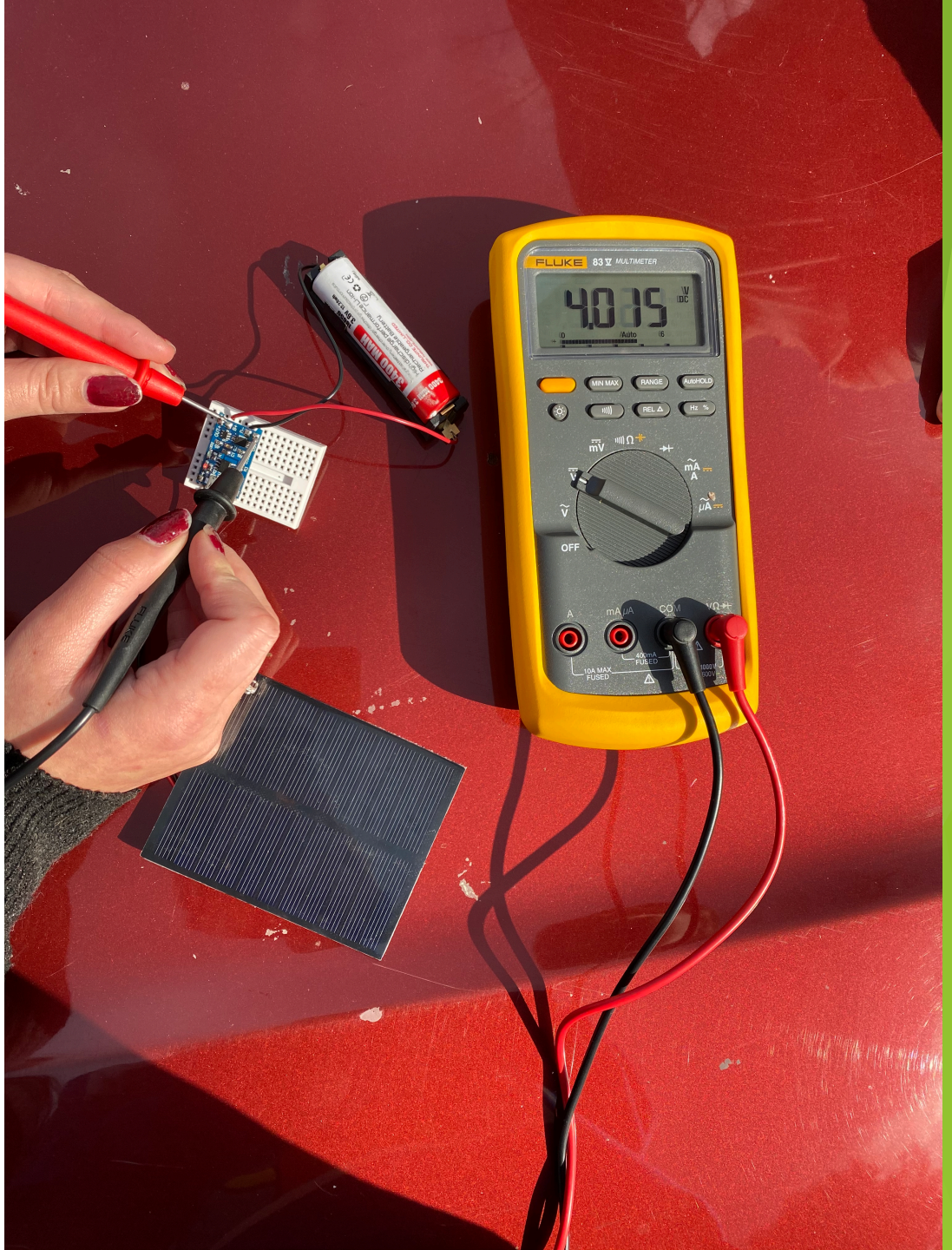
- ✓ Connection between solar panel and battery made
- ✓ Code configured to operate in low power modes
- ✓ **PRIORITY:** Framework of visual representation

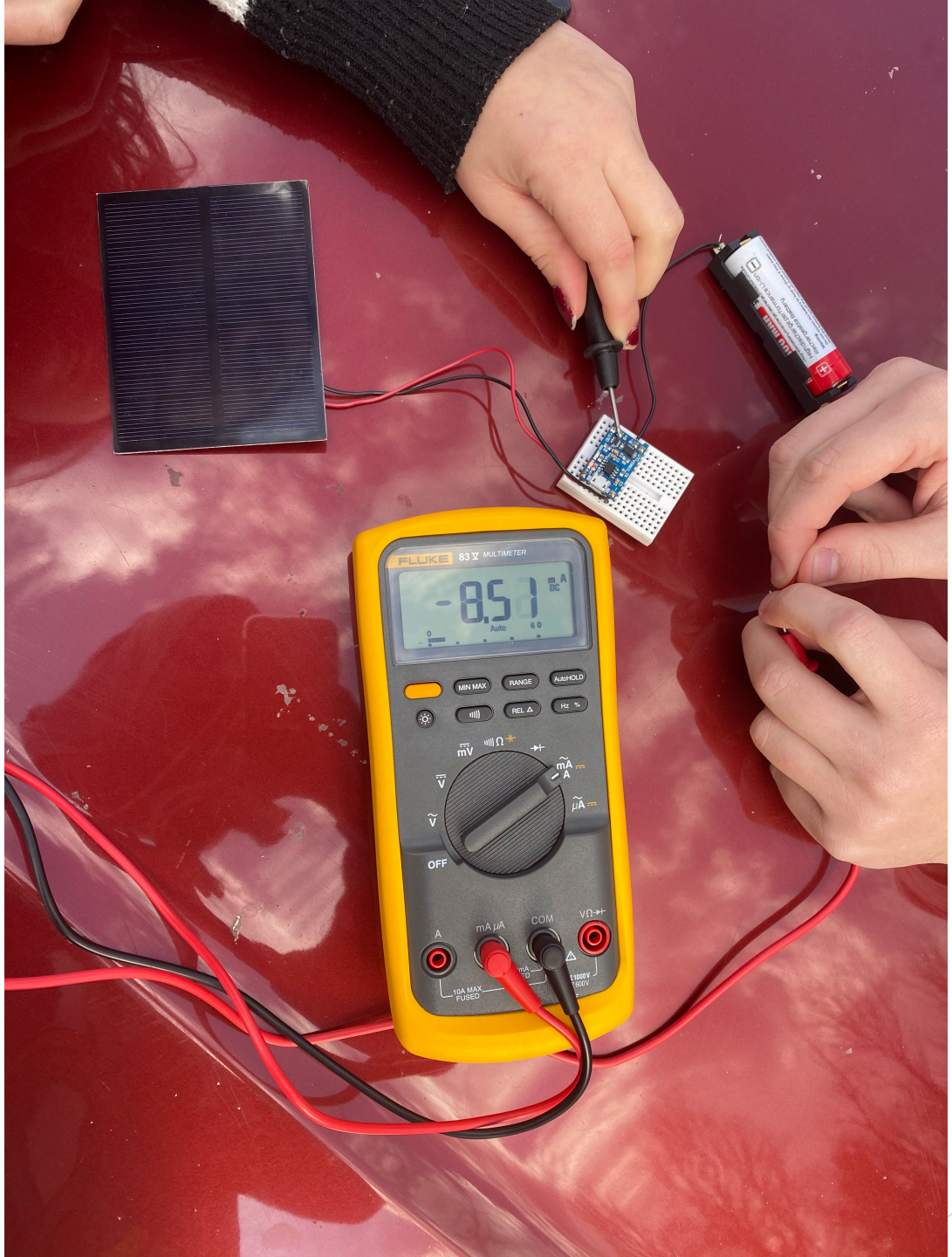
Demonstration

Power Demonstration Video




FLUKE 83 V MULTIMETER
4.458
OFF


FLUKE 83 V MULTIMETER
4.015
OFF


FLUKE 83 V MULTIMETER
-8.51
mA DC
Auto
MIN MAX
RANGE
Auto/HOLD
REL Δ
Hz %
OFF
A
mA μA
COM
VΩ
10A MAX FUSED

TensorFlow

```
[ ]  data_root = ("/content/drive/Shared drives/SPD21/Collab Notebook/data/train")
```

```
[ ]  #set image shape
     IMAGE_SHAPE = (224, 244) #(height,width) in no. of pixels

     #set the traiing data directory
     TRAINING_DATA_DIR = str(data_root)

     #rescale image and split data into training and validation
     datagen_kwargs = dict(rescale=1./255, validation_split=.20)

     #create train_generator and valid_generator and generating batches of tensor image data
     valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(**datagen_kwargs)
     valid_generator = valid_datagen.flow_from_directory(TRAINING_DATA_DIR,
                                                         subset="validation",
                                                         shuffle=True,
                                                         target_size=IMAGE_SHAPE)
     train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(**datagen_kwargs)
     train_generator = train_datagen.flow_from_directory(TRAINING_DATA_DIR,
                                                         subset="training",
                                                         shuffle=True,
                                                         target_size=IMAGE_SHAPE)

     #output - first line is validation data, second line is training data
```

```
Found 757 images belonging to 6 classes.
Found 3042 images belonging to 6 classes.
```

Setting up dataset

Dataset is split into 80% training and 20% validation

Images are shuffled

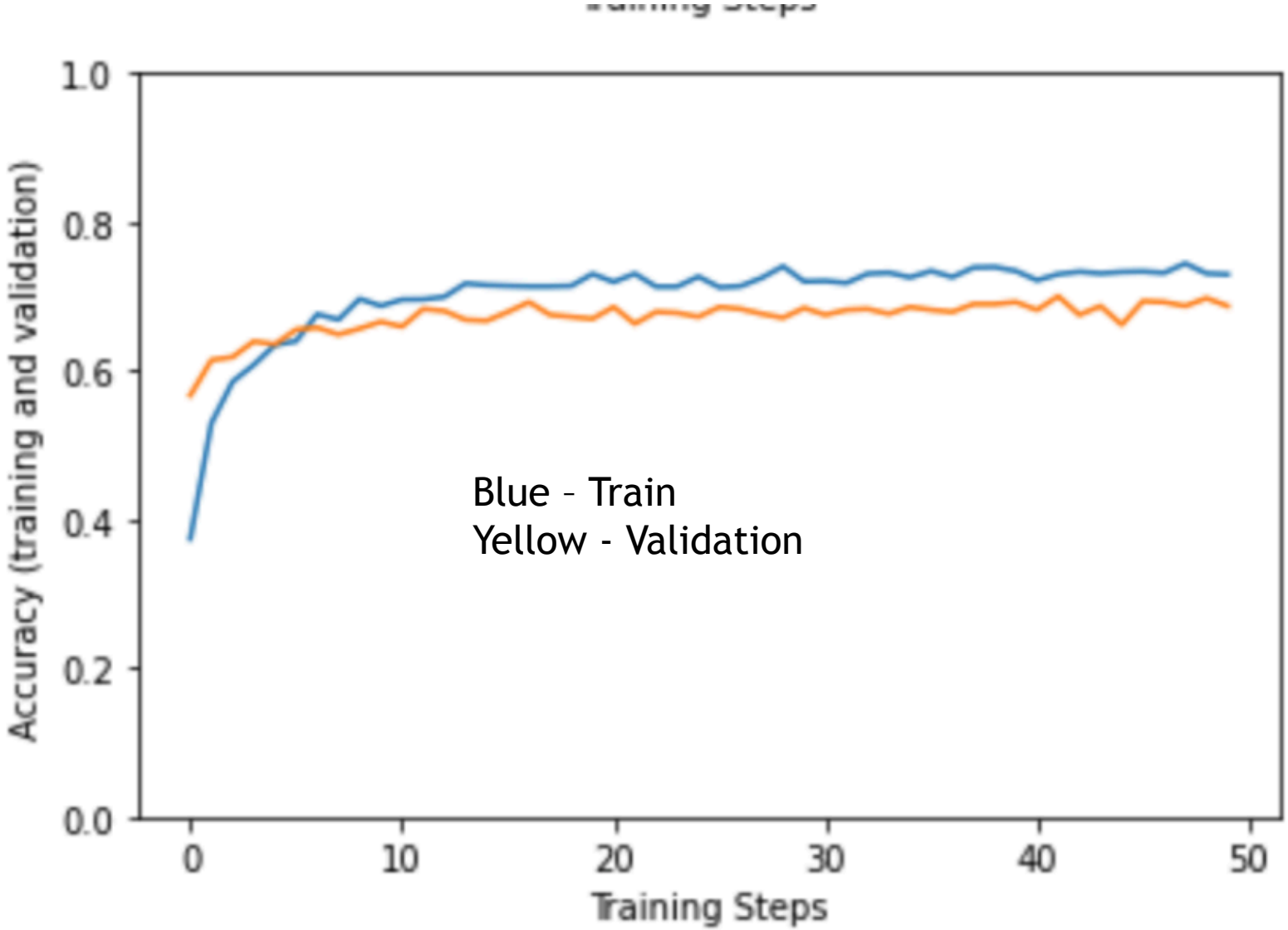
```
#As a base model for transfer learning, we'll use MobileNet v2 model stored on TensorFlow Hub.
#This model has advantages to be able to work on Mobile applications.
#For more information: https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4
model = tf.keras.Sequential([
hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4",
output_shape=[1280],
trainable=False),
tf.keras.layers.Dropout(0.4),
tf.keras.layers.Dense(train_generator.num_classes, activation='softmax')
])
model.build([None, 224, 224, 3])
model.summary()
model.compile(
optimizer=tf.keras.optimizers.Adam(),
loss='categorical_crossentropy',
metrics=['acc'])
```

Sequential Model
Base model - MobileNet v2 model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 7686
=================================================================
Total params: 2,265,670
Trainable params: 7,686
Non-trainable params: 2,257,984
_________________________________________________________________
```

Final Accuracy and Final Loss

```
[10] #check model
     final_loss, final_accuracy = model.evaluate(valid_generator, steps =
     print("Final loss: {:.2f}".format(final_loss))
     print("Final accuracy: {:.2f}%".format(final_accuracy * 100))

     24/24 [==============================] - 26s 1s/step - loss: 0.9245
     Final loss: 0.92
     Final accuracy: 72.95%
```


Accuracy (training and validation)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Training Steps
Blue – Train
Yellow - Validation

```
] #Testing our Model
  tf_model_predictions = insect_model.predict(val_image_batch)
  tf_pred_dataframe = pd.DataFrame(tf_model_predictions)
  tf_pred_dataframe.columns = dataset_labels
  print("Prediction results for the first elements")
  tf_pred_dataframe.head()
```

Prediction results for the first 5 images

```
Prediction results for the first elements
```

	Asiatic Rice Borer	Paddy Steam Maggot	Rice Gall Midge	Rice Leaf Caterpillar	Rice Leaf Roller	Yellow Rice Borer
0	0.73876303	0.00092524	0.00037819	0.00355539	0.19322839	0.06314977
1	0.39939523	0.00017391	0.00083112	0.25806096	0.01302752	0.32851130
2	0.01395532	0.00106388	0.00541841	0.24622387	0.72750521	0.00583335
3	0.33429050	0.00632006	0.01753991	0.46355417	0.15197450	0.02632090
4	0.88645256	0.02833018	0.02434595	0.01810730	0.01860134	0.02416258

Model predictions (green: correct, red: incorrect)

Converting Tensorflow to Tensorflow Lite Model

```
# Get the concrete function from the Keras model.
run_model = tf.function(lambda x : insect_model(x))
# Save the concrete function.
concrete_func = run_model.get_concrete_function(
tf.TensorSpec(model.inputs[0].shape, model.inputs[0].dtype)
)
# Convert the model
converter = tf.lite.TFLiteConverter.from_concrete_functions([concrete_func])
converted_tflite_model = converter.convert()
open(TFLITE_MODEL, "wb").write(converted_tflite_model)
# Convert the model to quantized version with post-training quantization
converter = tf.lite.TFLiteConverter.from_concrete_functions([concrete_func])
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
tflite_quant_model = converter.convert()
open(TFLITE_QUANT_MODEL, "wb").write(tflite_quant_model)
print("TFLite models and their sizes:")
!ls "tflite_models" -lh
```

```
TFLite models and their sizes:
total 11M
-rw-r--r-- 1 root root 2.3M Nov 18 21:45 insect_quant.tflite
-rw-r--r-- 1 root root 8.6M Nov 18 21:45 insect.tflite
```

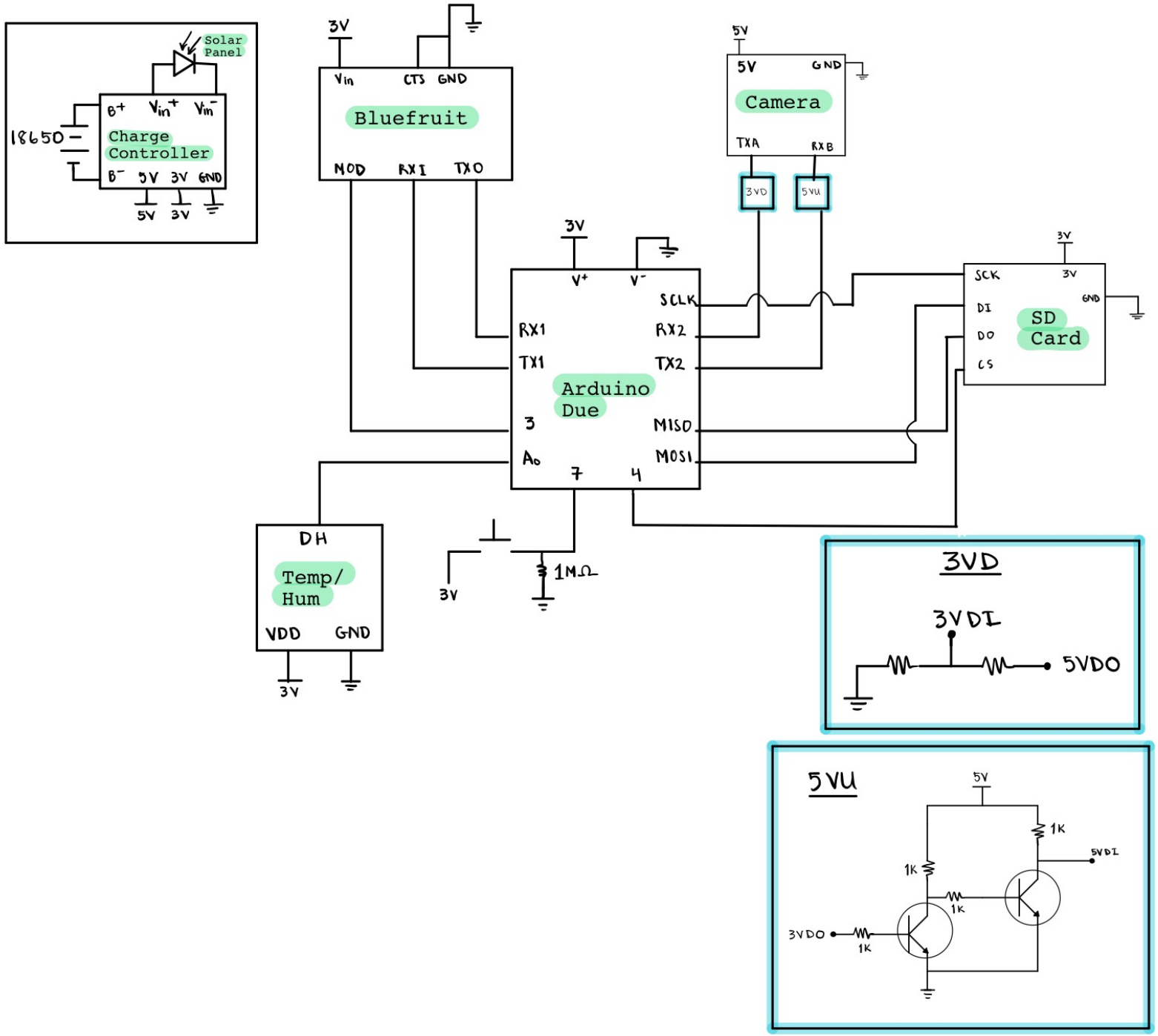
```
name: x
shape: [ 32 224 224   3]
type: <class 'numpy.float32'>

== Output details ==
name: Identity
shape: [32  5]
type: <class 'numpy.float32'>
[[0.00338608 0.0597314  0.7660391  0.14627957 0.02456389]
 [0.00338608 0.0597314  0.7660391  0.14627957 0.02456389]
 [0.00338608 0.0597314  0.7660391  0.14627957 0.02456389]
 [0.00621158 0.03705494 0.3135664  0.5243137  0.11885351]
 [0.00338608 0.0597314  0.7660391  0.14627957 0.02456389]]
pi@raspberrypi:~/ML2/examples/lite/examples/image_classification/raspbe
2020-11-18 19:36:02.353813: E tensorflow/core/platform/hadoop/hadoop_fi
== Input details ==
name: x
shape: [ 32 224 224   3]
type: <class 'numpy.float32'>

== Output details ==
name: Identity
shape: [32  5]
type: <class 'numpy.float32'>
(array([3, 2, 1, 0, 4], dtype=int32), array([[0.00338608, 0.0597314 , 0.7660391 , 0.14627957, 0.02456389],
       [0.00338608, 0.0597314 , 0.7660391 , 0.14627957, 0.02456389],
       [0.00338608, 0.0597314 , 0.7660391 , 0.14627957, 0.02456389],
       [0.00621158, 0.03705494, 0.3135664 , 0.5243137 , 0.11885351],
       [0.00338608, 0.0597314 , 0.7660391 , 0.14627957, 0.02456389]],
      dtype=float32))
pi@raspberrypi:~/ML2/examples/lite/examples/image_classification/raspberry_pi $
```

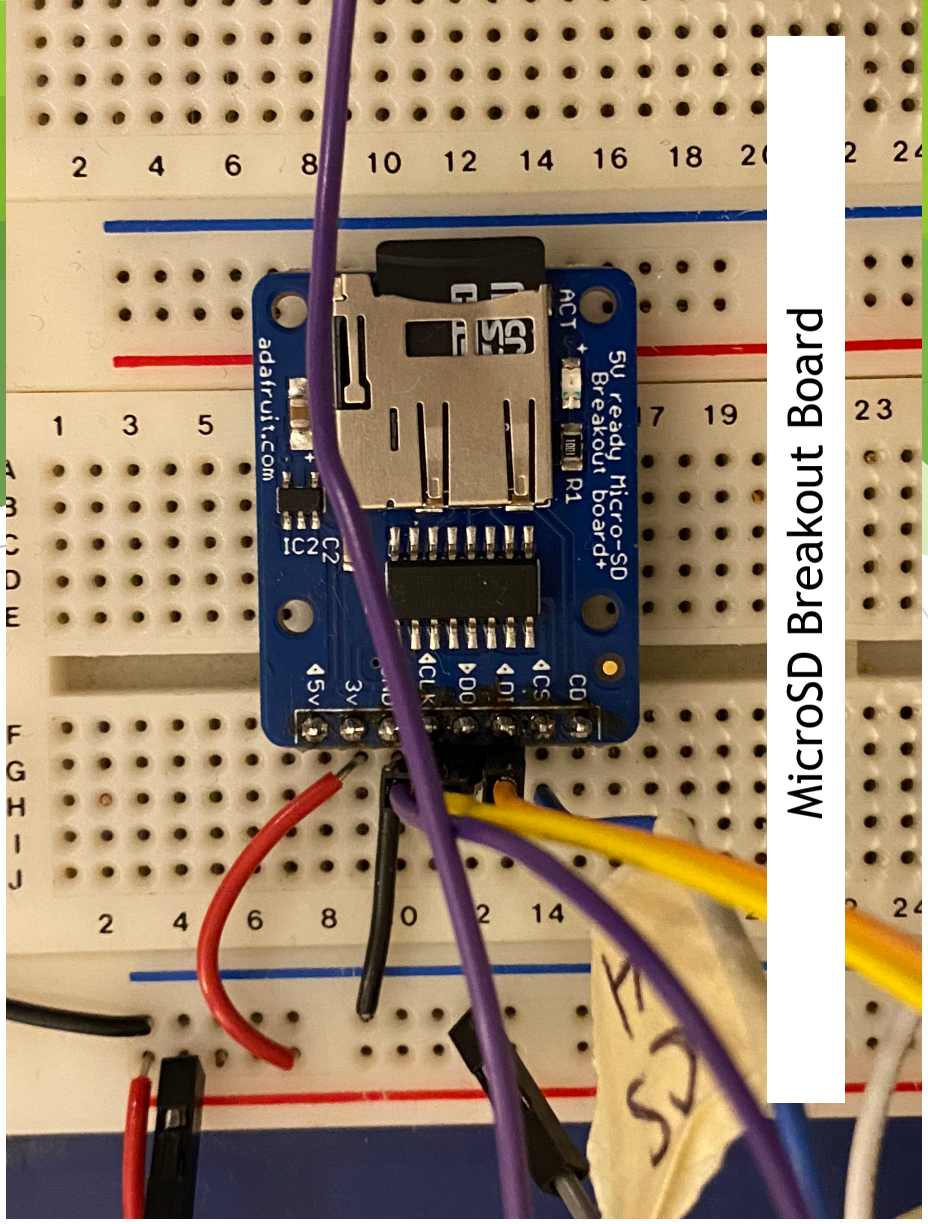
We can submit an image to raspberry pi and analyze the image using our TFLite model and output a probability table. We tried to let it print out like this.

Prediction results for the first elements

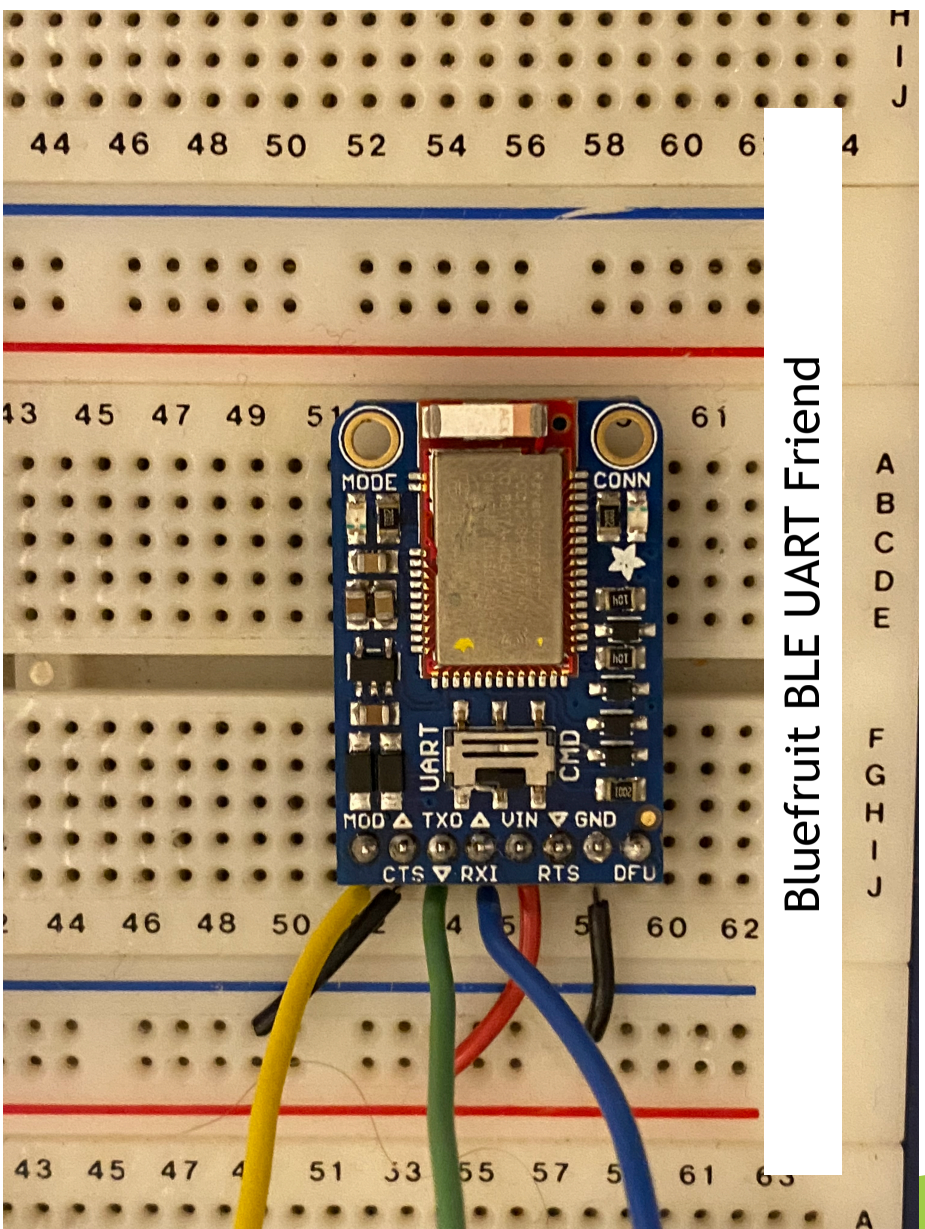
	Asiatic Rice Borer	Paddy Steam Maggot	Rice Gall Midge
0	0.73876303	0.00092524	0.00037819
1	0.39939523	0.00017391	0.00083112
2	0.01395532	0.00106388	0.00541841
3	0.33429050	0.00632006	0.01753991
4	0.88645256	0.02833018	0.02434595

- The camera we are using runs better off of a 5v supply, and the microcontroller is a 3v device, so we needed to circuitry to be able to shift the 3v to 5v.
- We added a MOSFET to restart the camera from the microcontroller in order to save power.
- This is also helped with the camera since it sometimes becomes unresponsive after running for too long.



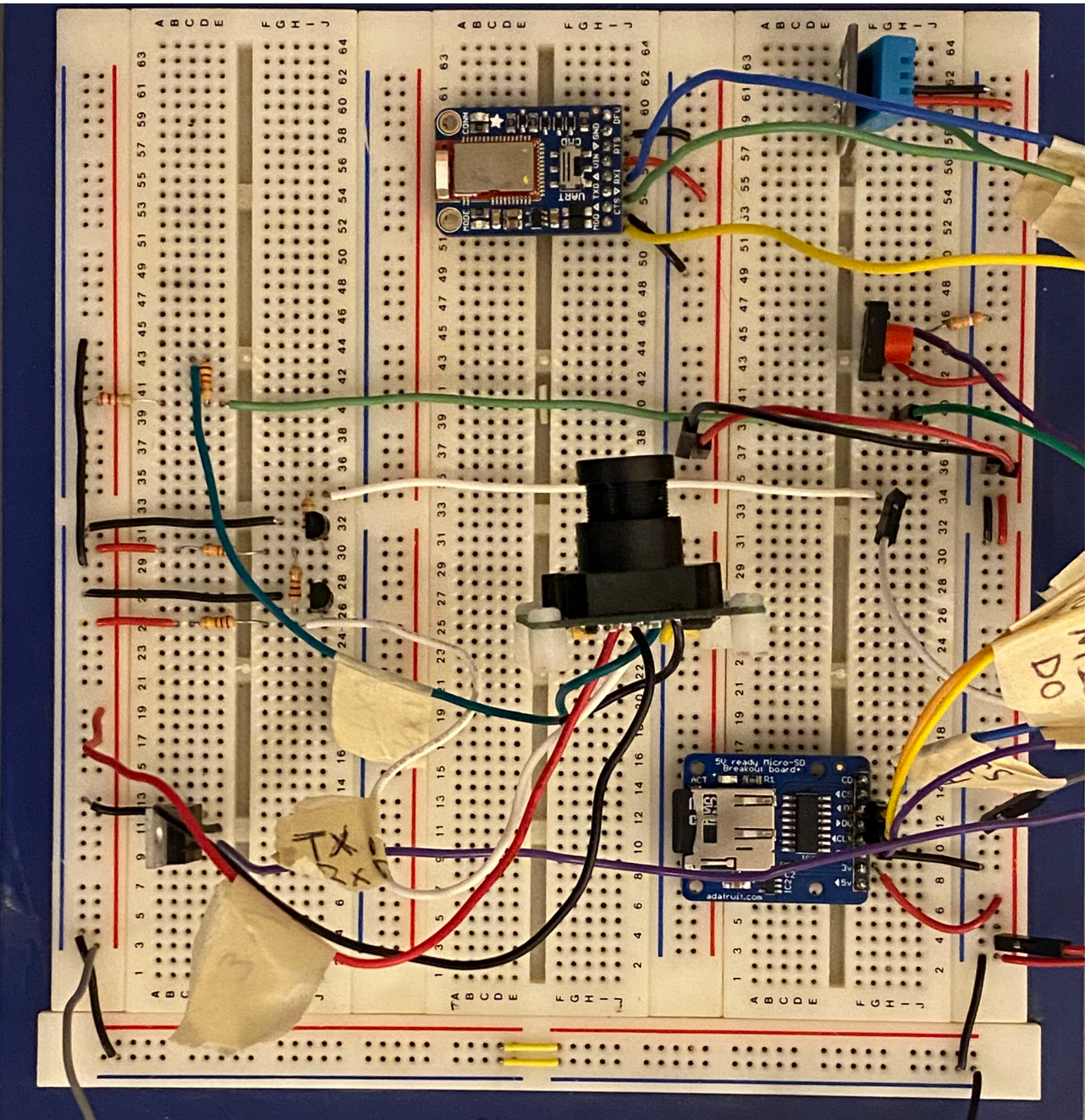


MicroSD Breakout Board

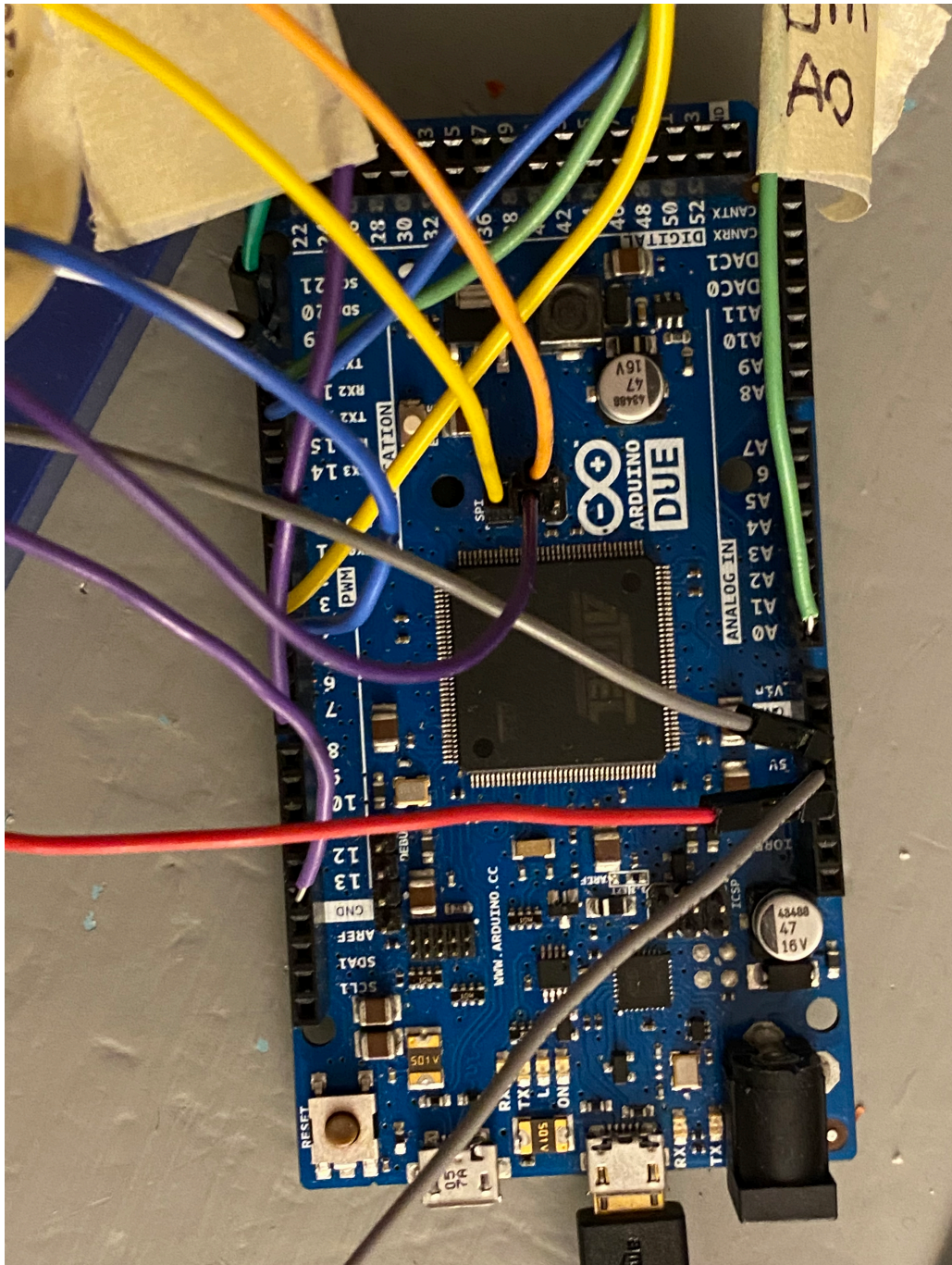


Bluefruit BLE UART Friend

Breadboard Circuit



Arduino Due



Camera and Data Conditioning Notes

Image

- Image size is VGA (640x480 pixels)
  - (320x240) and (160x120) are also options and require less transmission time, but we opted for better quality pictures
  - Output format: JPEG

Camera Connection

- UART

MicroSD Card

- Our microcontroller only has 2KB of data memory, and images are approximately 50KB
  - Needed to add the microSD card to store images before we can transfer them

Bluetooth Notes

Data Transmission Rate

- Currently, data can only be sent at approximately .5KBps
  - 4 Byte transmitted at a time with an 8ms delay between transmissions
  - Can transmit up to 16 Bytes at a time, but with increased data loss

Maximum Transmission Rate

- The nRF51822 we are using should be able to transfer at 3KBps, so we should be able to increase the speed.

File Size

- Images are compressed JPGs, around 50Kb, so it takes about 100 seconds to send an image.
- Text is sent more reliably, because it is considerably smaller

Sublime Text File Edit Selection Find View Goto Tools Project Window Help Fri Nov 20 10:28 AM

app.py — UNREGISTERED

app.py

```python
from flask import Flask, request,render_template
import sqlite3

app = Flask(__name__)

@app.route('/')
def home():
    conn = sqlite3.connect('pest_alert.db')
    cursor =conn.cursor()
    cursor.execute("SELECT * FROM info")
    data = cursor.fetchall()
    return render_template('template.html', data=data)

if __name__ == 'main':
    app.run(port=80, host = '192.168.1.250')

```

Line 1, Column 1 Tab Size: 4 Python

template.html — UNREGISTERED

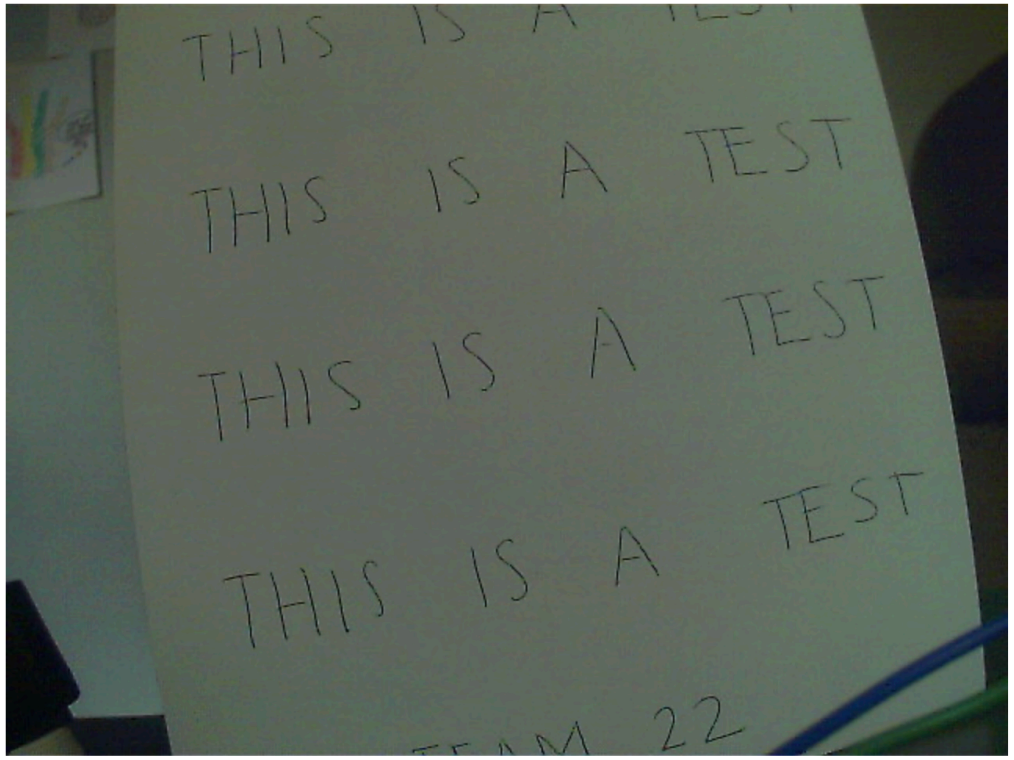
template.html

```html
<!DOCTYPE html>
<html>
<head>
 <title>Pest Alert</title>
</head>
<body>
<style>

 td {
        width: 150px;
        text-align: center;
        border: 1px solid black;
        padding: 5px;
      }
</style>
<table>
    <thead>
    <tr>
        <th>Picture</th>
        <th>RH Percentage</th>
        <th>Temperature in Degrees Celcius</th>
    </tr>
    </thead>
    <tbody>
        {% for row in data %}
            <tr>
                <td> <img src={{ url_for('static', filename=row[0]) }} alt="{{row[0]}}"> </
                td>
                <td>{{row[1]}}</td>
                <td>{{row[2]}}</td>
            </tr>
        {% endfor %}
    </tbody>
</table>
</body>
</html>
```

Line 1, Column 1 Spaces: 4 HTML

WebApp Preview

192.168.1.250

Picture	RH Percentage	Temperature in Degrees Celcius
	56.0	19.0

MDR Accomplishments

Madelyn

- Connection between solar panel and battery made
- Code configured to operate in low power modes
- Framework of visual representation

Aisha

- The temperature and humidity sensor have been configured and are functioning
- Camera is configured; can capture images when user-requested

Owen

- BLE Communication is established and data conditioning is successful

Angela

- Database configured
- 70% accuracy in bug detection

List of Hardware and Software

Current

- Adafruit MicroSD Breakout Board
- Adafruit Bluefruit BLE UART Friend
- TC4056A Battery Charging Module
- 18650 Cell Battery
- 1W Solar Panel
- TTL Serial Camera Module
- DHT11 Temperature and Humidity Sensor Module
- Raspberry Pi 3B+
- Arduino Due
- Tensorflow
- Flask/Python
- SQLite

Projected

- MicroSD Card Socket
- nRF51822 BLE Module
- TC4056A Battery Charging IC
- 18650 Cell Battery
- 1W Solar Panel
- TTL Serial Camera Module
- Honeywell Temperature and Humidity Sensor IC
- Raspberry Pi 3B+
- Microchip SAM L10
- Tensorflow
- Flask/Python
- SQLite

FPR Hardware Plan

- MicroSD Card Socket
- nRF51822 Bluetooth Low Energy Module
- TC4056A Battery Charging IC
- 18650 Cell Battery
- Honeywell SENS HUMI/TEMP
- Transistors for shifting between 3V and 5V
- MOSFET for camera power management

- 1W Solar Panel
- TTL Serial Camera Module

Hardware Expenditures

Current:

Item	SPARKFUN Serial Basic Breakout	EDGE Development Board	Lithium Battery	HIMAX Camera	SAM L10 Board	Solar Panel	Motion Sensor	Battery Holder	BLE Chip SMD
Price	$8.95	$18.69	$5.95	$10.00	$59.16	$12.74	$6.08	$3.93	$10.45

Item	Charging IC	Humidity/Temp Sensor	Adafruit MicroSD Breakout Board	Adafruit BLE	Shipping	Current Total
Price	$8.55	$12.96	$8.06	$18.14	$7.99	$191.65

Projected:

Item	nRF51BLE	SAM L10 SOIC	Micro SD Card Socket	PCB Printing	Projected Total	Overall Total
Price	$7.95	$1.98	$3.00	$15.00	$27.93	$219.58

Project Management

Team Responsibilities:

- Team Coordinator: Madelyn Wright
- Budget Management Lead: Angela Wong
- Altium Lead: Owen Boucher
- Technical Responsibilities: Aisha Ben-Neticha

Date Start:	2/1/2021													
				Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Task	Start	End	Team Member	Feb 1 - 6	Feb 7 - 13	Feb 14 - 20	Feb 21 - 27	Feb 28 - Mar 6	Mar 7 - 13	Mar 14 - 20	Mar 21 - 27	Mar 28 - Apr 3	Apr 4 - 10	Apr 11 - 17	Apr 18 - 24
Website	Feb 1st	Feb 6th	Aisha												
Design PCB	Feb 1st	Feb 13th	Owen												
Coding for Battery	Feb 1st	Feb 13th	Madelyn												
Order PCB	Feb 14th	Feb 27th	Owen												
Web App	Feb 21st	March 6th	Angela												
CDR Week	March 8th	March 12th	ALL												
Possibly replace SD board	March14th	March 27th	Aisha												
Possibly replace BLE board	March 21st	April 3rd	Owen												
Switch Wiring to PCB	April 4th	April 17th	ALL												
FPR Week	April 19th	April 23rd	ALL												

Questions?