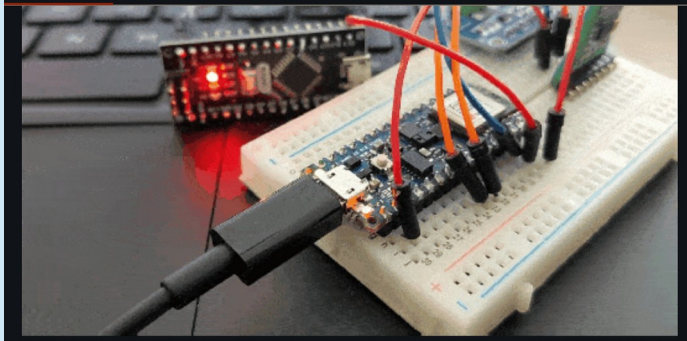
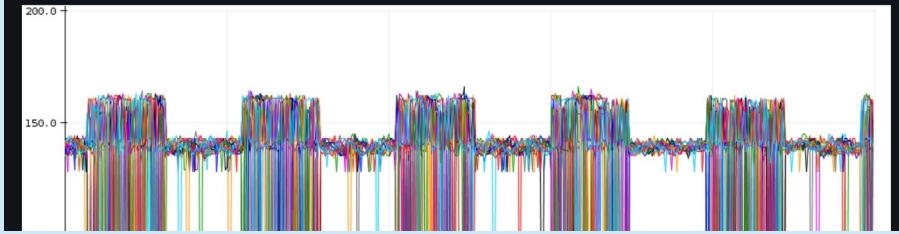
Inspiration

README MIT license



What is CurrentSense-TinyML (and does it work?)

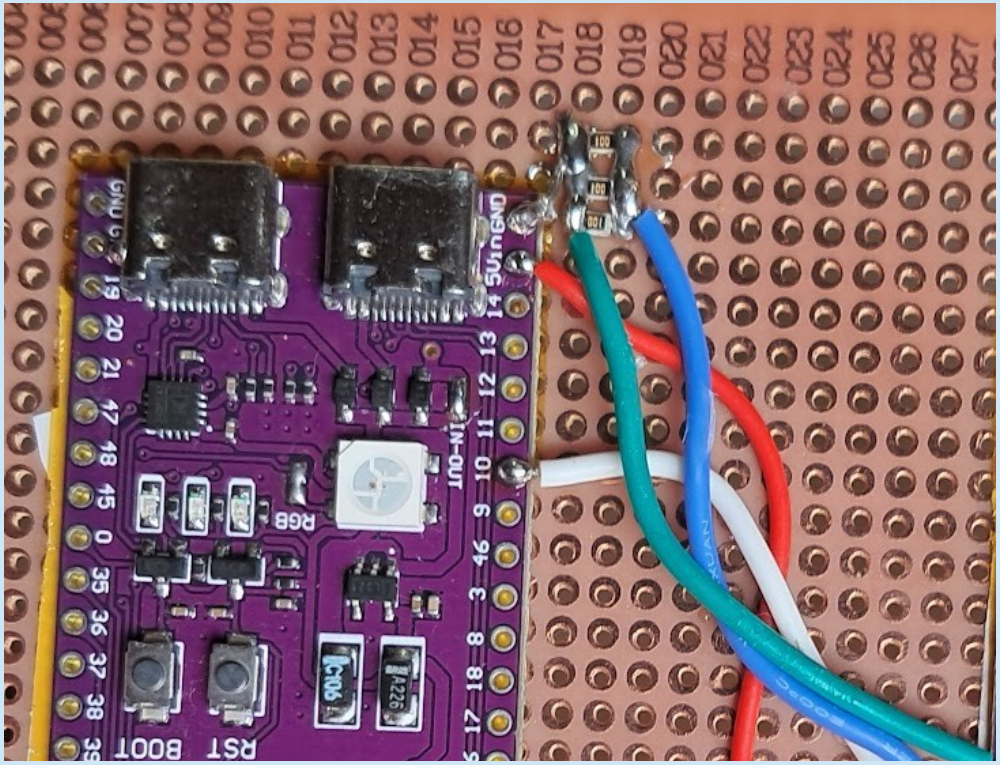
Despite prior evidence that says 'yes!' from the work we cited above, there are a few good indicators that this should work. If we setup the INA219 with the Nano 33 Sense and just monitor the Nano target running blink, we can see the following output when we use Arduino IDE's Serial Plotter (using the get_current_data.ino code for those who want to play along at home)



- Inspiration
  - Santander Group Cyber Security Research Team
  - Read current from target and predict LED on/off state
  - https://github.com/Santandersecurityresearch/CurrentSense-TinyML

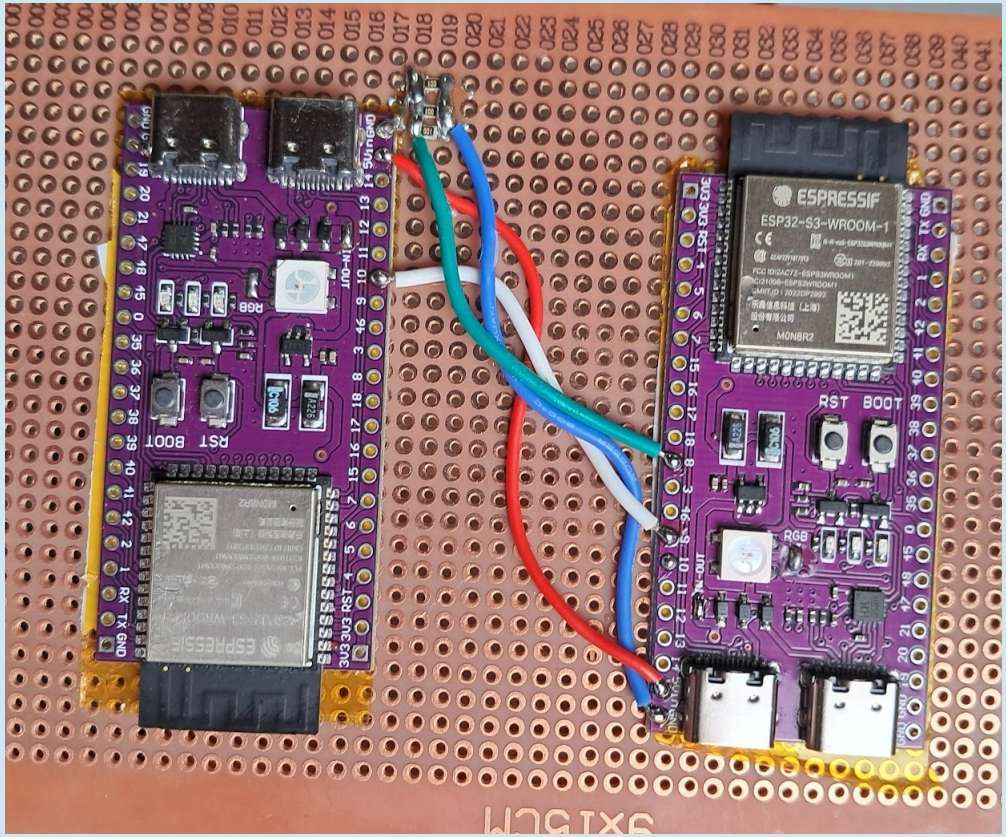
01 June 2024, ESTGA, Águeda

Hardware



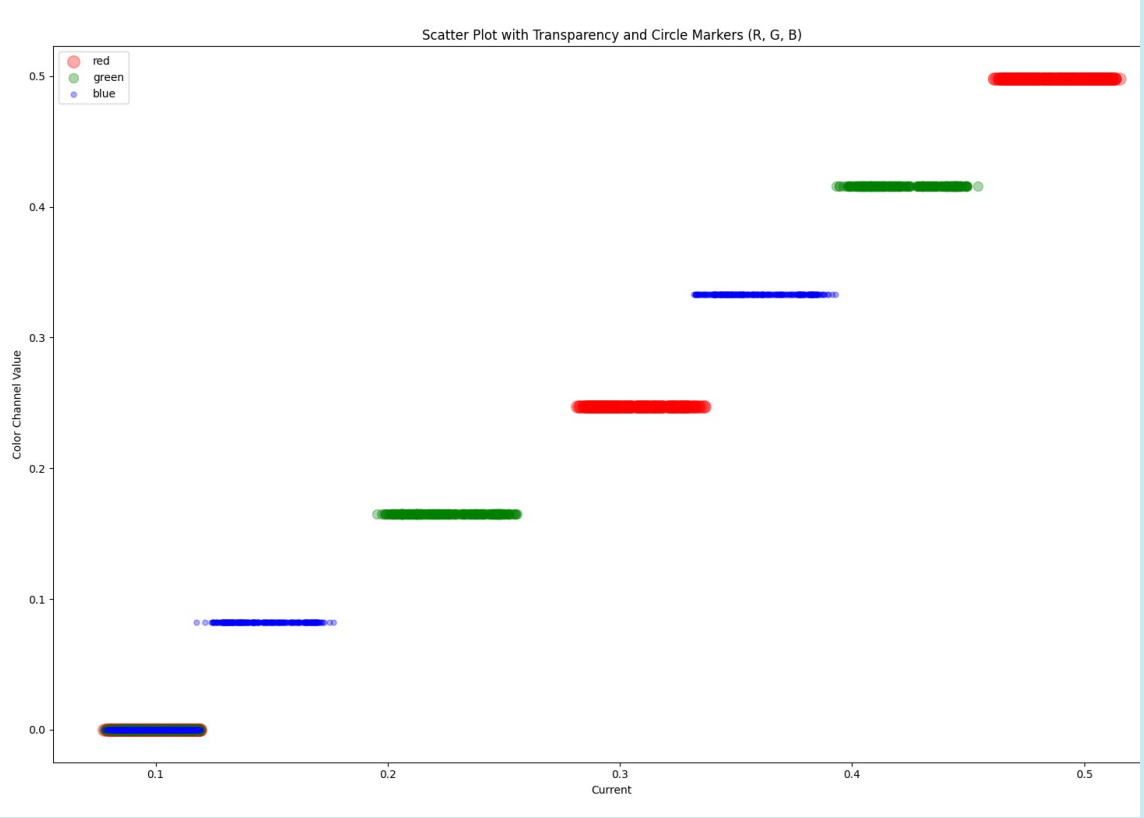
- Target board
  - 3 different energy values for each R, G, B channels??
- Observer board
- Poor man current reading: ADC + resistor
- Python firmware

Training



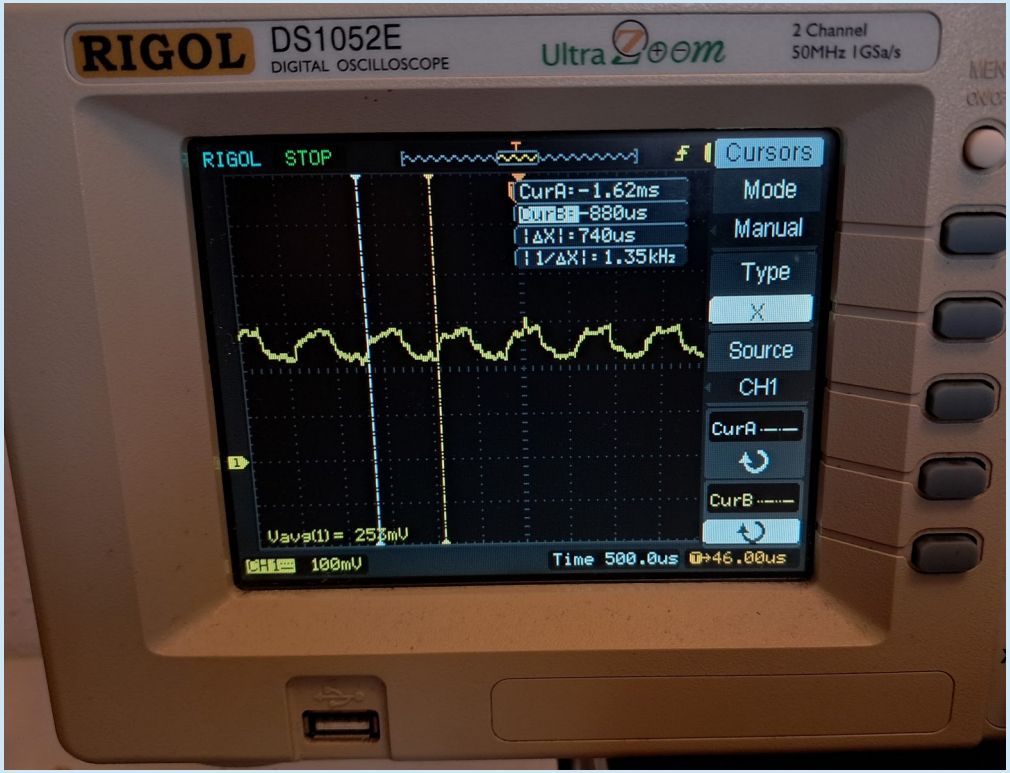
- Target board
  - Random blink RGB LED
- Observer board
  - Read target board energy usage
- Target board → RGB state → Observer board
- Observer board → RGB state and energy → PC

To much noise



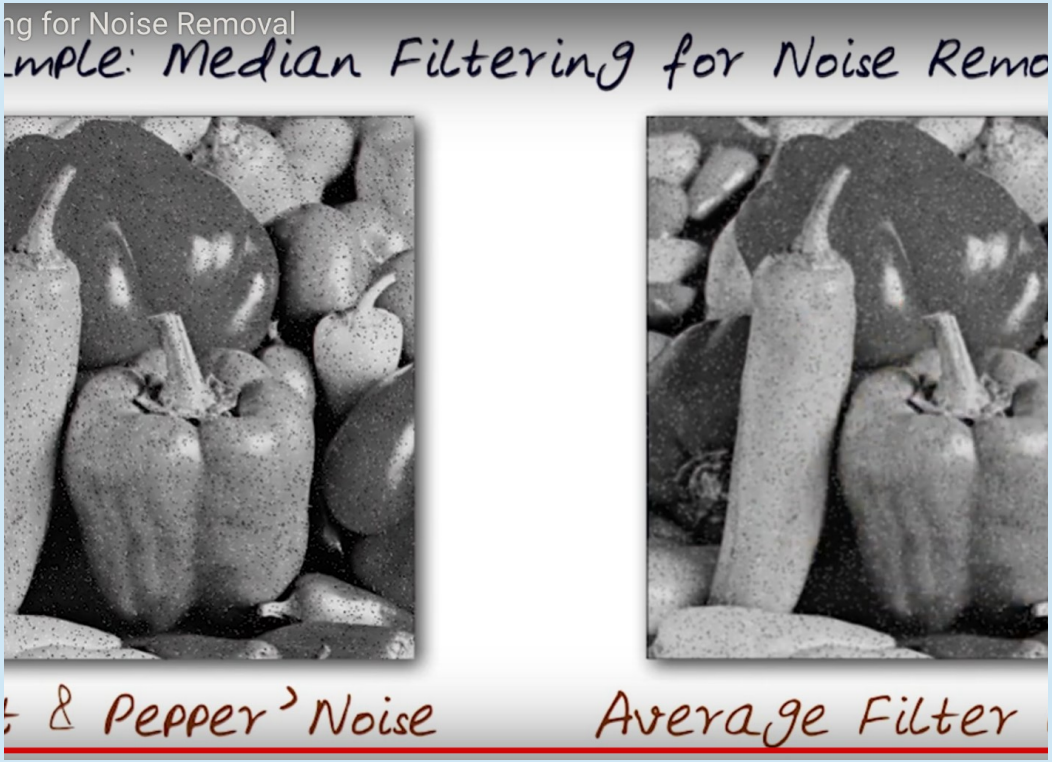
- Values overlap due to to much noise
- Possible actions
  - Check the electric signal
  - Use a dedicated IC for measurements
  - Testing software filters

PWM energy



- IC controlling each R G B channels
  - Energy is equal for each channel
  - Pulsating energy, not constant

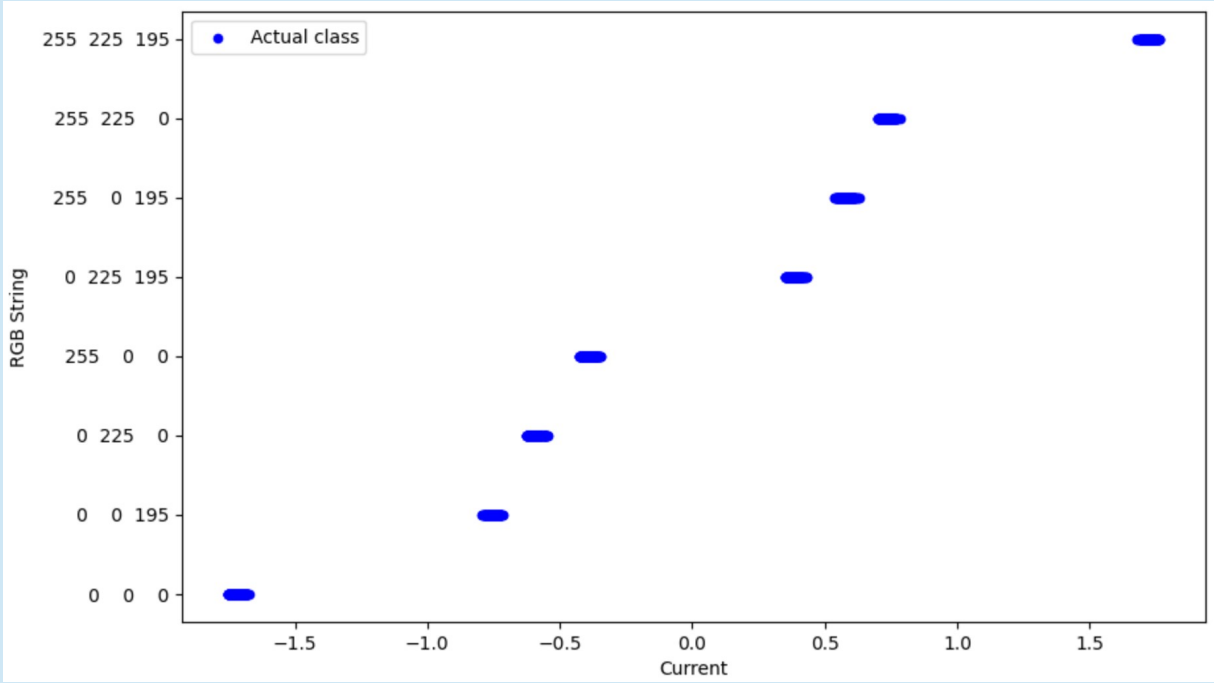
Filters



https://www.youtube.com/watch?v=lcfSk9RP8xA

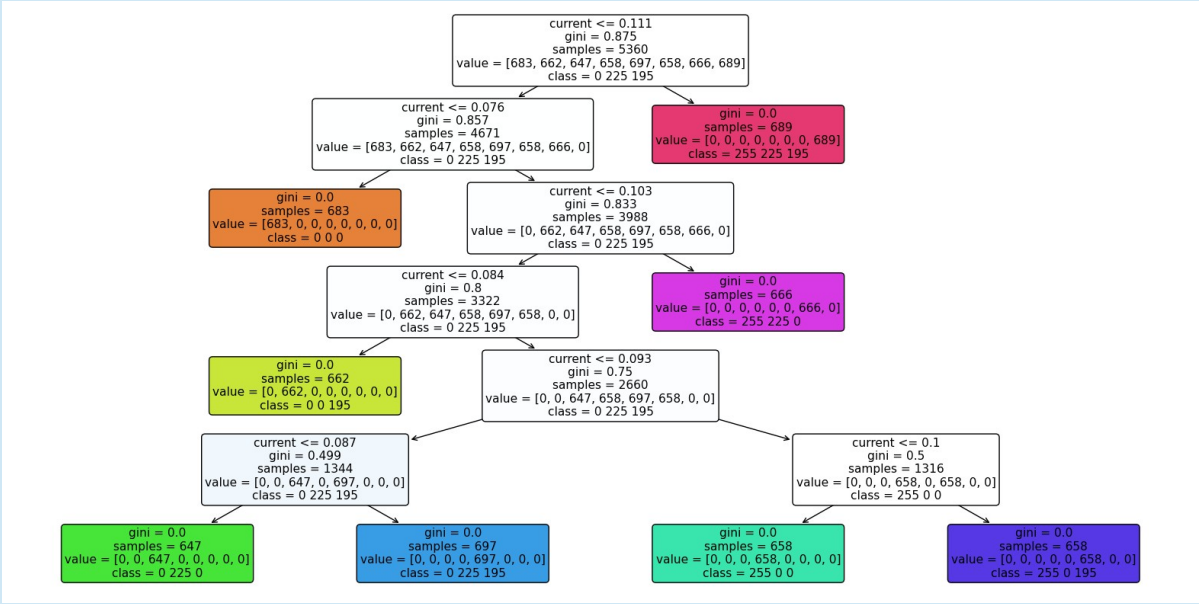
- PWM filtering: reading signal 5 or 10x slower
- Noise filtering: median of 15 readings
  - 1/5 of a second
  - STD measure to evaluate Median VS AVG

Filters (cont.)



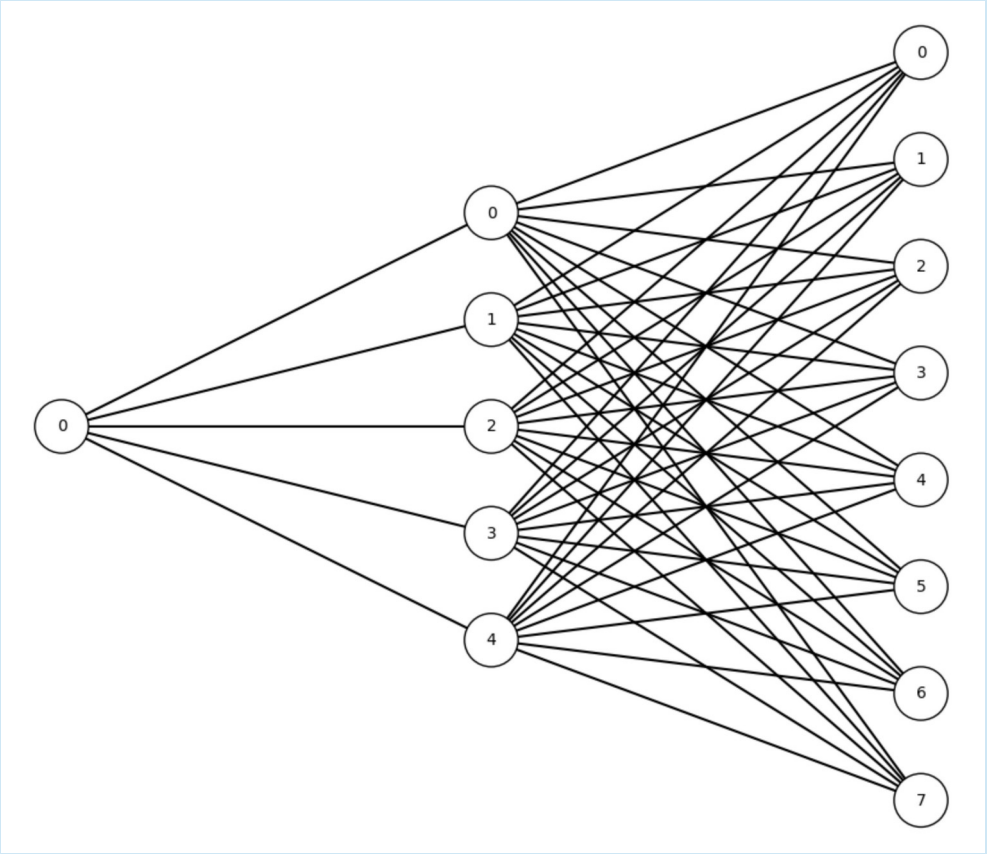
- After
  - Dedicated IC for measurements
  - Software filters

Decision Tree



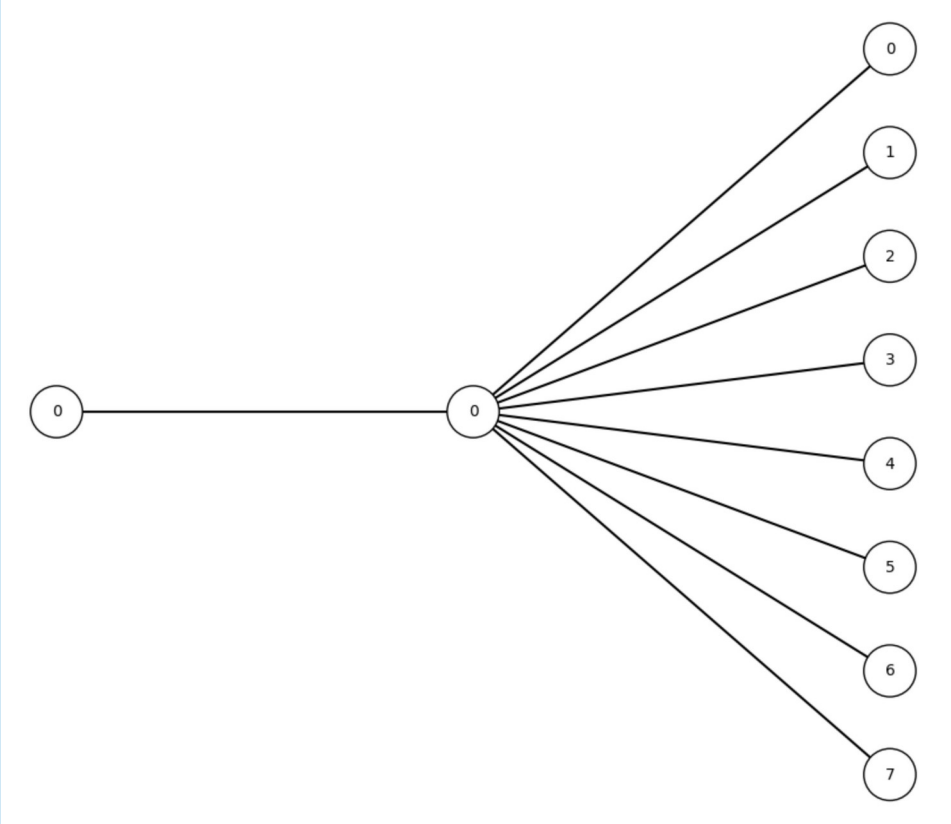
- Precision 100%, “perfect” confusion matrix
- sklearn.tree plot_tree
- Very simple to implement
- Implemented on the microcontroller
- Processing time: ~1us

Simple NN



- Precision 100%, “perfect” confusion matrix
- 1 hidden layer
- 5 nodes only

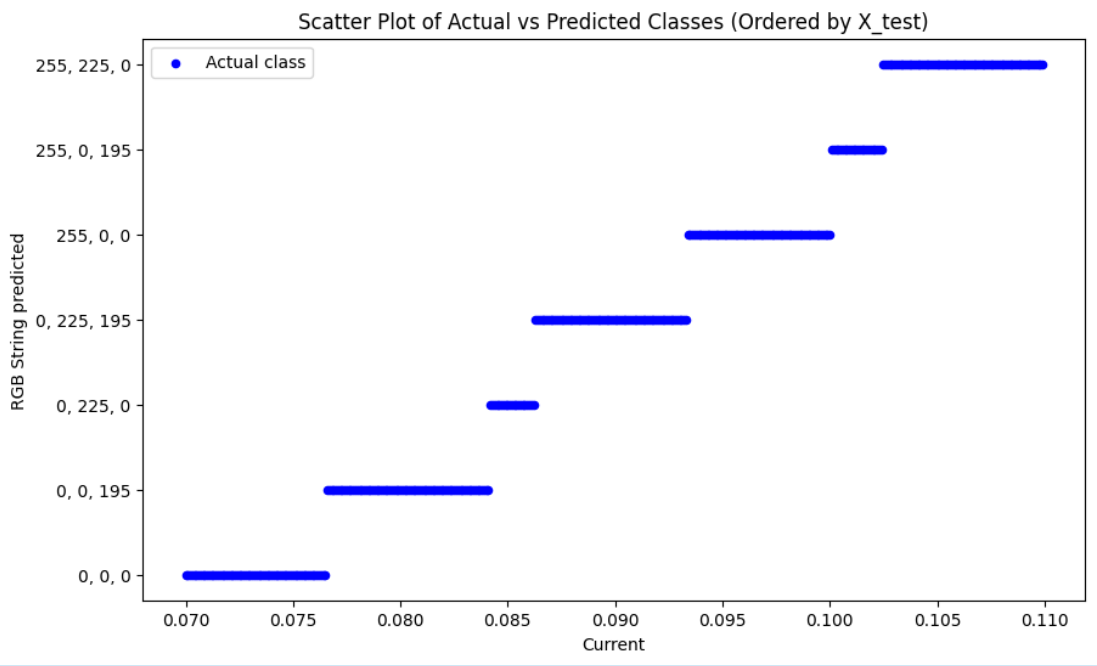
Simple NN (cont.)



- Precision 100%, “perfect” confusion matrix
- 1 hidden layer, 1 node only
- Weights, Biases exported from sklearn
- Processing time: 500us
- ESP32-S3 has matrix multiplication DSP

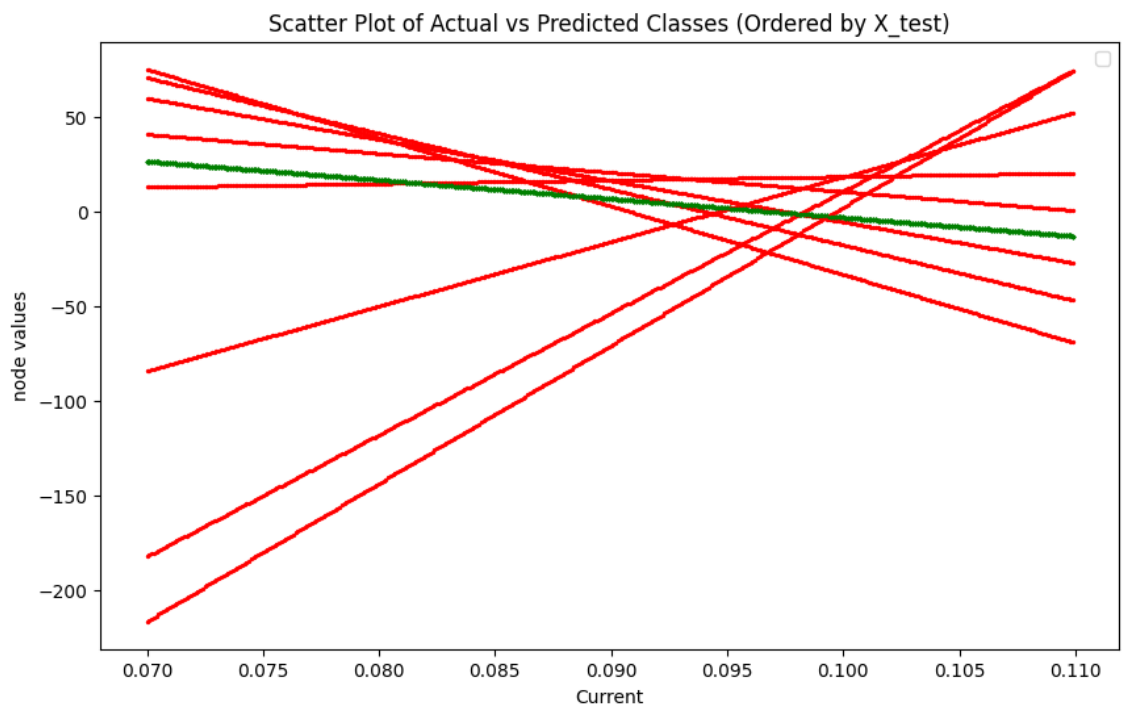
```
def predict(self, X):
  layer = X
  weights = self._weights.copy()
  biases = self._biases.copy()

  for layer_index in range(self._num_layers):
    layer = np.dot(layer, weights[layer_index]) + biases[layer_index]

    # using identity activation function, so no need to call the activa
    # layer = self.relu(layer)

  node_number = np.argmax(layer.tolist())
  return self._classes[node_number]
```

Simple NN (cont.)



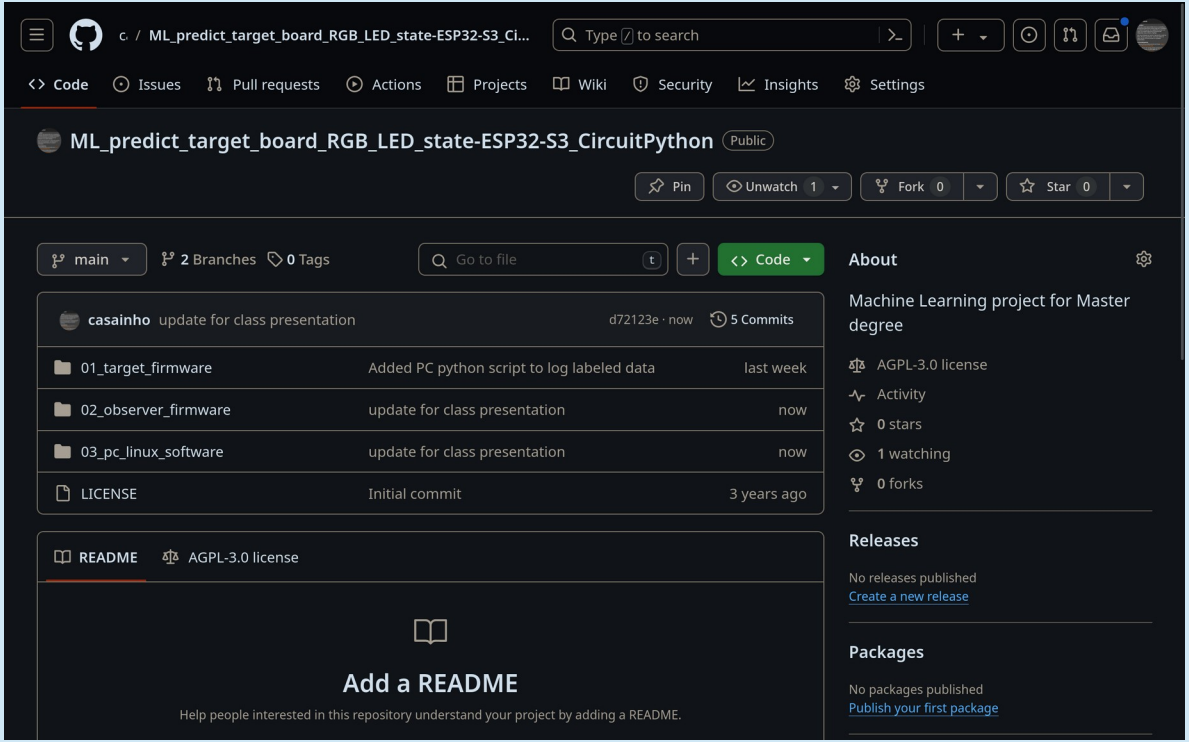
- Input all possible valus and check the transitions between classes

Simple NN (cont.)



- Checked the internal states
- Each node has identify activation function
  - Y = (weight * X) + bias
  - Straight line equation
- Each line represents the value on the output node
- Node with MAX value → class

Future work



- Is possible to predict the next RGB value by predicting the next RANDOM value generated inside the microcontroller?