



Using Keyword Spotting to Control Physical Systems

Ezenia Diaz-Lembert & Gage Hills
Harvard University

CRESTLEX3 Workshop, 2021

<https://tinymlx.org/CRESTLEX3/>



So Far

Human
Intelligence

Artificial
Intelligence

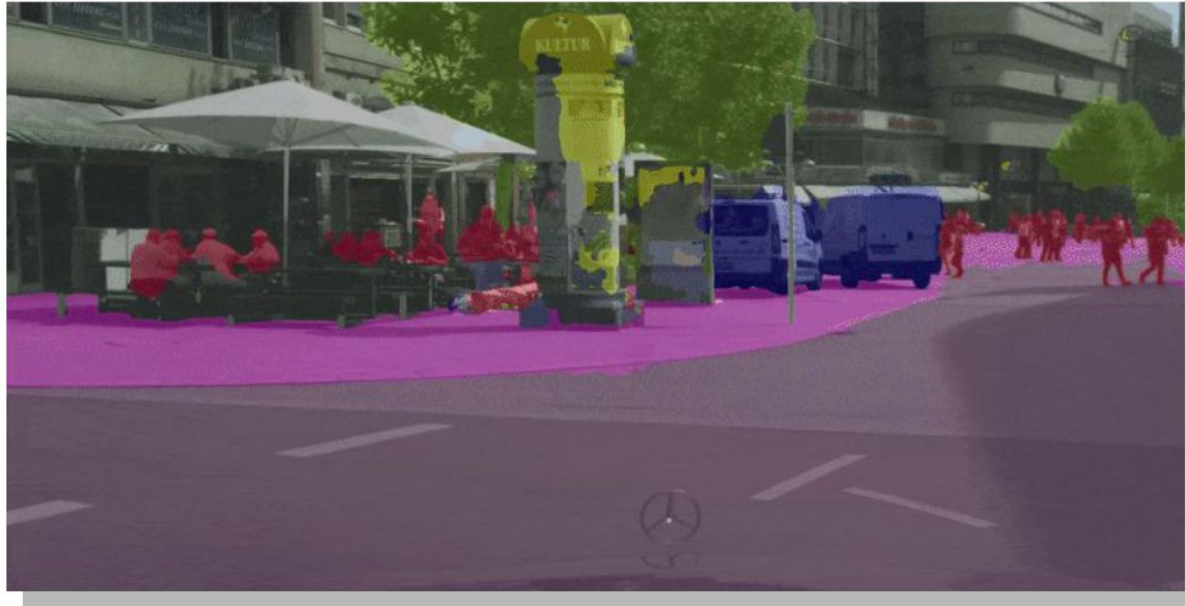
Classification

Vision

Audio

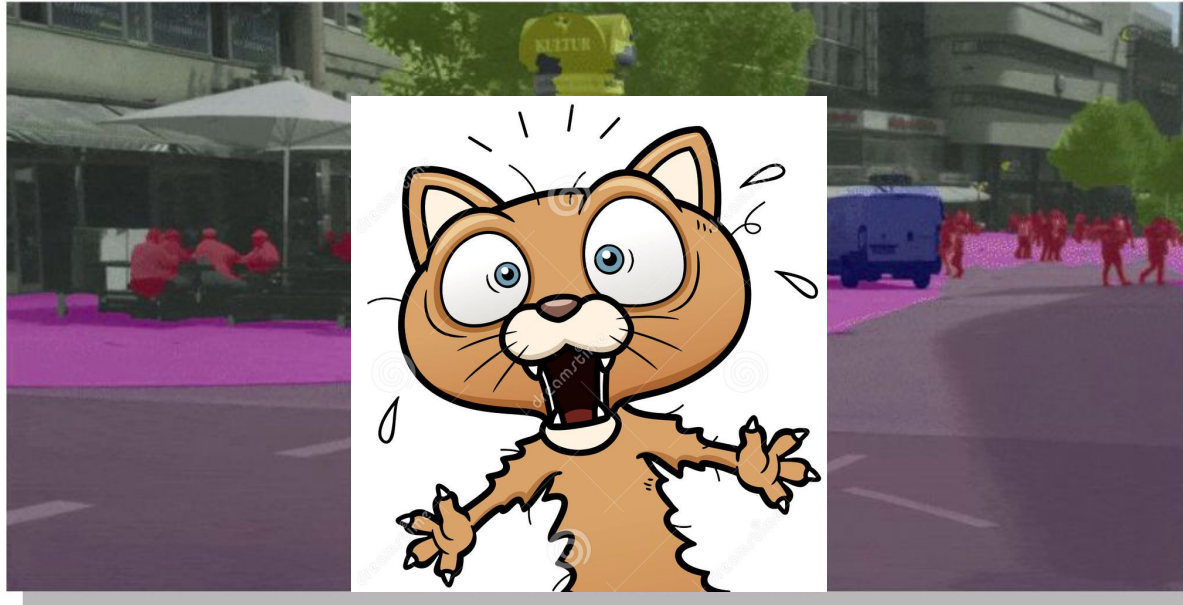
So Far

Segmentation



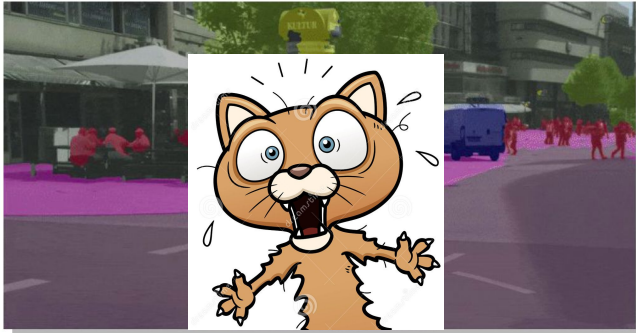
So Far

“Cat” detected



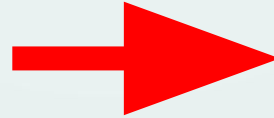
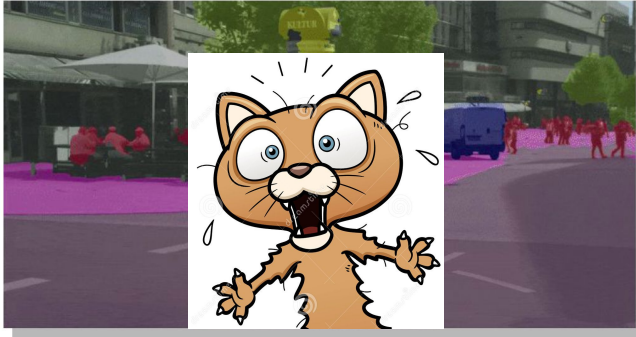
Then What ?

Cat detected...



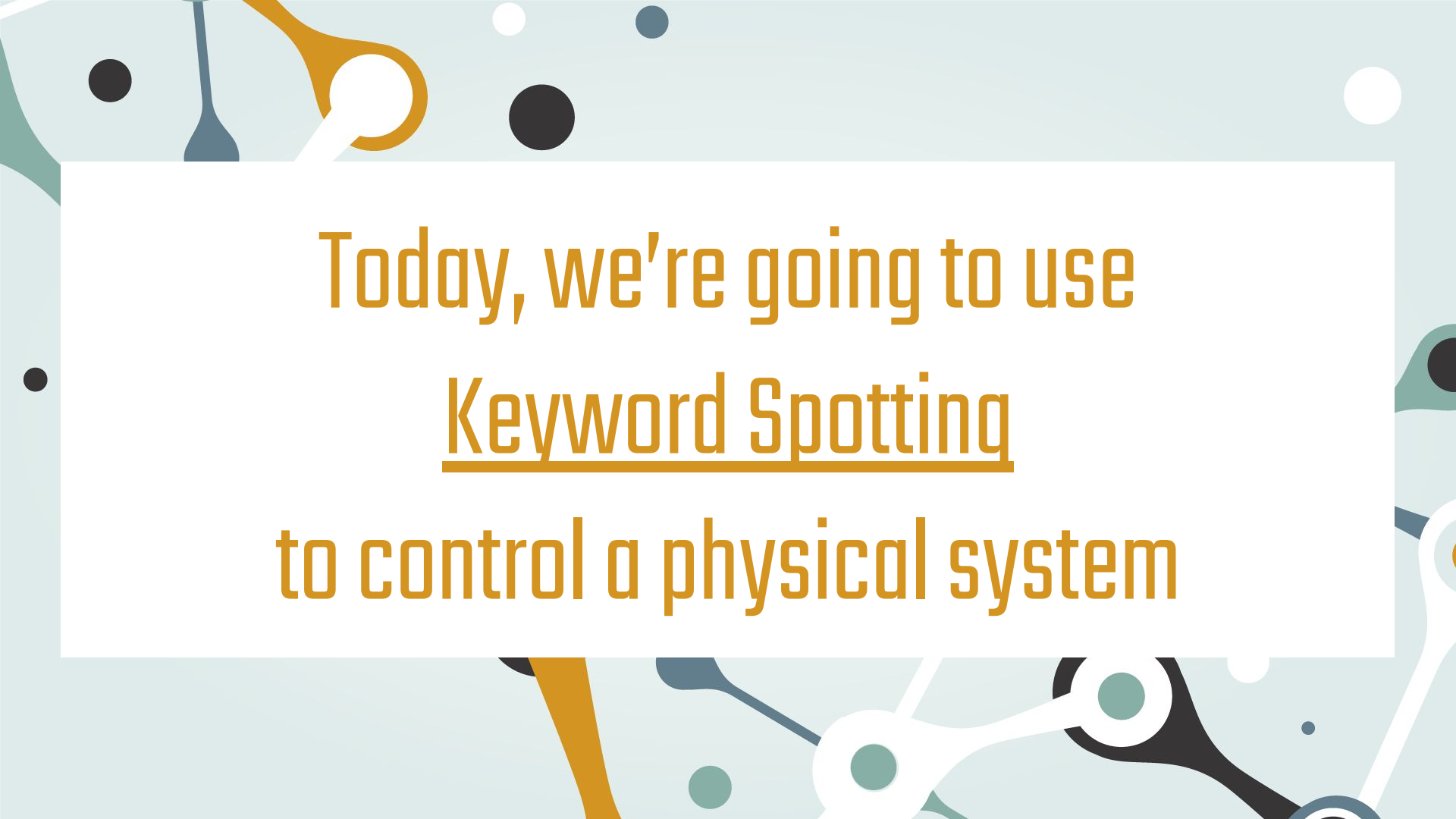
Then What ?

Cat detected...



BRAKE!!!





Today, we're going to use
Keyword Spotting
to control a physical system

The Automatic Pet Feeder



In particular, we're going to be "hacking" into an automatic pet feeder!

This feeder is made to rotate at set times to feed pets! So... we know a couple things:

- Some power is driving the feeding container to move (motor)
- Some programming is being used to control this device!

This will be incredibly useful for our hacking!

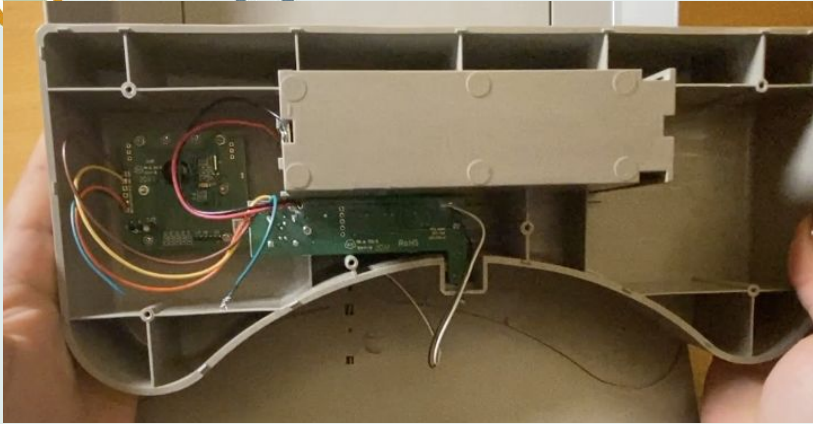
The Automatic Pet Feeder



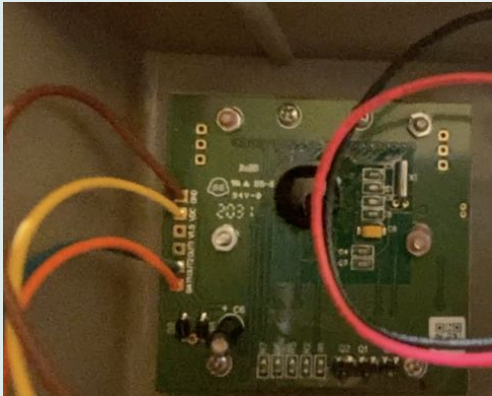
Once you have opened the feeder, it should look like this!

- Here we can see the removable container (if you take it off, you can see the motor), and control system of the pet feeder
- But, how does the pet feeder know **when** to rotate ? How does it know how to rotate at all?

How Does it Work?



- After unscrewing the back of the pet feeder, you should see something like this beneath the control system of the feeder
- Consists of boards taking into account the digital signals and programming set by the user from the buttons, and the control of power to the motor



Today

Step 1:

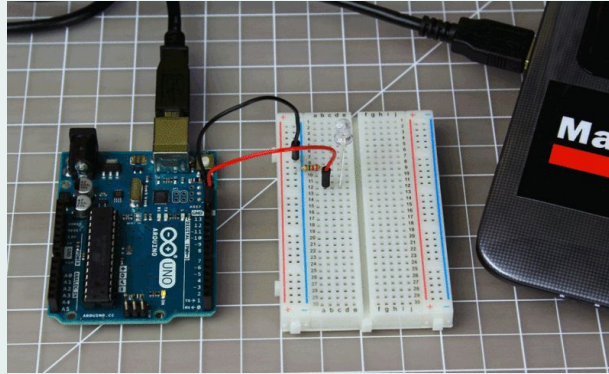
Build TinyML model



EDGE IMPULSE

Step 2:

Run it on Hardware



Arduino BLE Nano 33

Step 3:

Feed my cats



Pet feeder

Step 1: Build TinyML Model

The screenshot shows the Edge Impulse web interface. On the left is a navigation sidebar with the following items: Dashboard, Devices, Data acquisition, Impulse design (with sub-items: Create impulse, MFCC, NN Classifier), Retrain model, Live classification, Model testing, Versioning, and Deployment. The main content area has a purple header with 'Project info', 'Keys', and 'Export' tabs, and a user profile for 'Gage Hills'. Below the header, the project name 'Gage Hills / cat' is displayed, followed by a description: 'This is your Edge Impulse project. From here you acquire new training data, design impulses and train models.' The main content is divided into three panels: 1. 'Creating your first impulse (100% complete)' which includes a step 'Acquire data' (describing data capture from a development board, phone, or import) with a 'LET'S COLLECT SOME DATA' button, and a step 'Design an impulse' (describing teaching the model to interpret new data). 2. 'Sharing' which states 'Your project is private' and has a 'Make this project public' button. 3. 'Summary' which shows 'DEVICES CONNECTED'.

EDGE IMPULSE

- Dashboard
- Devices
- Data acquisition
 - Create impulse
 - MFCC
 - NN Classifier
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment

Project info Keys Export Gage Hills

Gage Hills / cat

This is your Edge Impulse project. From here you acquire new training data, design impulses and train models.

Creating your first impulse (100% complete)

- Acquire data**
Every Machine Learning project starts with data. You can capture data from a development board or your phone, or import data you already collected.
[LET'S COLLECT SOME DATA](#)
- Design an impulse**
Teach the model to interpret previously unseen data, based on historical data. Use this to categorize new data, or to find anomalies in sensor

Sharing

Your project is private.

[Make this project public](#)

Summary

DEVICES CONNECTED

Step 1: Build TinyML Model

The screenshot displays the Edge Impulse web interface. On the left is a navigation sidebar with the following items: Dashboard, Devices, Data acquisition, Impulse design, Create impulse, MFCC, NN Classifier, Retrain model, Live classification, Model testing, Versioning, and Deployment. The main content area is titled "DATA ACQUISITION (CAT)" and includes a user profile for "Gage Hills". There are two tabs: "Training data" (selected) and "Test data". A notification banner reads: "Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options". Below this are three summary cards: "DATA COLLECT..." showing "1m 0s" with a bar chart icon, "LABELS" showing "4" with a pie chart icon, and "Record new data" with a "Connect using WebUSB" button. A message states "No devices connected to the remote management API." At the bottom right, a dark blue box says "RAW DATA Click on a sample to load...". The central "Collected data" table is as follows:

SAMPLE NAME	LABEL	ADDED	LENGTH	
kitty.28qk8m6...	kitty	Yesterday, ...	1s	⋮
kitty.28qk8m6...	kitty	Yesterday, ...	1s	⋮
kitty.28qk8m6...	kitty	Yesterday, ...	1s	⋮
kitty.28qk8m6...	kitty	Yesterday, ...	1s	⋮
kitty.28qk7m1...	kitty	Yesterday, ...	1s	⋮

Step 1: Build TinyML Model

The screenshot displays a software interface for editing audio samples. The main window is titled "Crop sample 'llama.28qkiot.s5'". The audio waveform is plotted on a dark blue background with a vertical axis ranging from -1500 to 1000. The horizontal axis shows sample indices from 0 to 9123. A white rectangular crop box is positioned over a segment of the waveform, spanning from approximately sample 8109 to 9123. In the top right corner of the editor, there is a control labeled "Set sample length (ms.):" with a dropdown menu currently set to "1000". At the bottom of the interface, there is a playback control bar showing "0:00 / 0:00" and a volume icon. The interface also includes a "Cancel" button on the bottom left and a "Crop" button on the bottom right.

Step 1: Build TinyML Model

EDGE IMPULSE

CREATE IMPULSE (CAT) Gage Hills

An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

- Time series data**
 - Axes: audio
 - Window size: 1000 ms.
 - Window increase: 500 ms.
 - Zero-pad data:
- Audio (MFCC)**
 - Name: MFCC
 - Input axes: audio
- Neural Network (Keras)**
 - Name: NN Classifier
 - Input features: MFCC
 - Output features: 4 (giraffe, kitty, llama, penguin)
- Output features**
 - 4 (giraffe, kitty, llama, penguin)


Save Impulse

Dashboard
Devices
Data acquisition
Impulse design

- Create impulse
 - MFCC
 - NN Classifier

Retrain model
Live classification
Model testing
Versioning
Deployment

Step 1: Build TinyML Model



- Dashboard
- Devices
- Data acquisition
- Impulse design
 - Create impulse
 - MFCC
 - NN Classifier
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment

MFCC (CAT) Gage Hills

Parameters [Generate features](#)

Training set

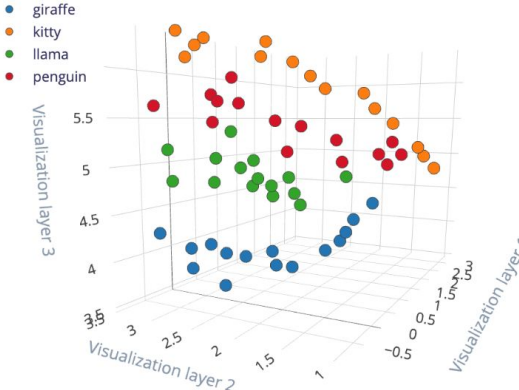
Data in training set	1m 0s
Classes	4 (giraffe, kitty, llama, penguin)
Window length	1000 ms.
Window increase	500 ms.
Training windows	60

[Generate features](#)

Feature explorer (60 samples)

X Axis: Visualization Y Axis: Visualization Z Axis: Visualization

● giraffe ● kitty ● llama ● penguin



Step 1: Build TinyML Model

The screenshot displays the Edge Impulse web interface for configuring an NN Classifier model. The interface is divided into a left sidebar with navigation options and a main content area with configuration panels.

EDGE IMPULSE

- Dashboard
- Devices
- Data acquisition
- Impulse design
 - Create impulse
 - MFCC
 - NN Classifier
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment

NN CLASSIFIER (CAT) Gage Hills

#1 [Click to set a description for this version](#)

Neural Network settings

Training settings

Number of training cycles

Learning rate

Minimum confidence rating

Audio training options

Data augmentation

Neural network architecture

Architecture presets [1D Convolutional \(Default\)](#)
[2D Convolutional](#)

Training output

Model Model version: [?](#) [Quantized \(int8\)](#)

Last training performance (validation set)

ACCURACY **75.0%** **LOSS** **0.54**

Confusion matrix (validation set)

	GIRAFFE	KITTY	LLAMA	PENGU...
GIRAFFE	100%	0%	0%	0%
KITTY	0%	100%	0%	0%
LLAMA	0%	66.7%	33.3%	0%
PENGUIN	0%	0%	25%	75%
F1 SCORE	1.00	0.75	0.40	0.86

Feature explorer (full training set) [?](#)

Step 2: Run it on Arduino

The screenshot shows the Edge Impulse web interface. On the left is a navigation sidebar with the following items: Dashboard, Devices, Data acquisition, Impulse design (with a sub-menu containing 'Create impulse', 'MFCC', and 'NN Classifier'), Retrain model, Live classification, Model testing, Versioning, and Deployment. The main content area has a purple header with 'DEPLOYMENT (CAT)' and a user profile for 'Gage Hills'. Below the header is a white card titled 'Deploy your impulse' with the text: 'You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)' Below this is a section titled 'Create library' with the text: 'Turn your impulse into optimized source code that you can run on any device.' There are five library options shown in a grid: 'C++ library' (with a C++ logo), 'Arduino library' (with an Arduino logo and a blue border), 'Cube.MX CMSIS-PACK' (with a Cube logo), 'WebAssembly' (with a WA logo), and 'TensorRT library' (with an NVIDIA logo).

EDGE IMPULSE

DEPLOYMENT (CAT) Gage Hills

Deploy your impulse

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

Create library

Turn your impulse into optimized source code that you can run on any device.

- C++ library
- Arduino library**
- Cube.MX CMSIS-PACK
- WebAssembly
- TensorRT library

Step 2: Run it on Arduino

The screenshot displays the Edge Impulse web interface. On the left is a navigation sidebar with the following items: Dashboard, Devices, Data acquisition, Impulse design, Create impulse, MFCC, NN Classifier, Retrain model, Live classification, Model testing, Versioning, and Deployment. The main content area is titled 'options.' and features a toggle for 'Enable EON™ Compiler' with the text 'Same accuracy, up to 50% less memory. Open source.' To the right, a 'Build output' section shows the progress of a build job (ID: 1015061) with logs including 'SDK...', 'SDK OK', and 'updating headers...'. A large white modal window is centered on the screen, displaying a green checkmark icon and the text: 'Built Arduino library', 'Add this library through the Arduino IDE via:', 'Sketch > Include Library > Add .ZIP Library...', 'Examples can then be found under:', and 'File > Examples > cat_inferencing'. A green 'Build' button is visible at the bottom of the modal.

Step 2: Run it on Arduino

```
nano_ble33_sense_microphone | Arduino 1.8.15
nano_ble33_sense_microphone $
/* Includes ----- */
#include <PDM.h>
#include <cat_inferencing.h>

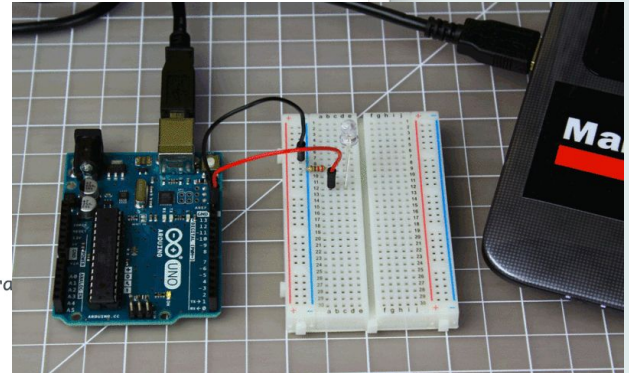
/** Audio buffers, pointers and selectors */
typedef struct {
    int16_t *buffer;
    uint8_t buf_ready;
    uint32_t buf_count;
    uint32_t n_samples;
} inference_t;

static inference_t inference;
static signed short sampleBuffer[2048];
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw

/**
 * @brief      Arduino setup function
 */
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);

    Serial.println("Edge Impulse Inferencing Demo");

```



Arduino BLE Nano 33

Step 2: Run it on Arduino

```
void setup()
{
  // put your setup code here, to run once:
  Serial.begin(115200);

  Serial.println("Edge Impulse Inferencing Demo");

  // summary of inferencing settings (from model_metadata.h)
  ei_printf("Inferencing settings:\n");
  ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
  ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
  ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
  ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) / size

  if (microphone_inference_start(EI_CLASSIFIER_RAW_SAMPLE_COUNT) == false) {
    ei_printf("ERR: Failed to setup audio sampling\r\n");
    return;
  }

  // ghills
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(2, OUTPUT);
}
```

Step 2: Run it on Arduino

```
nano_ble33_sense_microphone | Arduino 1.8.15

nano_ble33_sense_microphone $
ei_impulse_result_t result = { 0 };

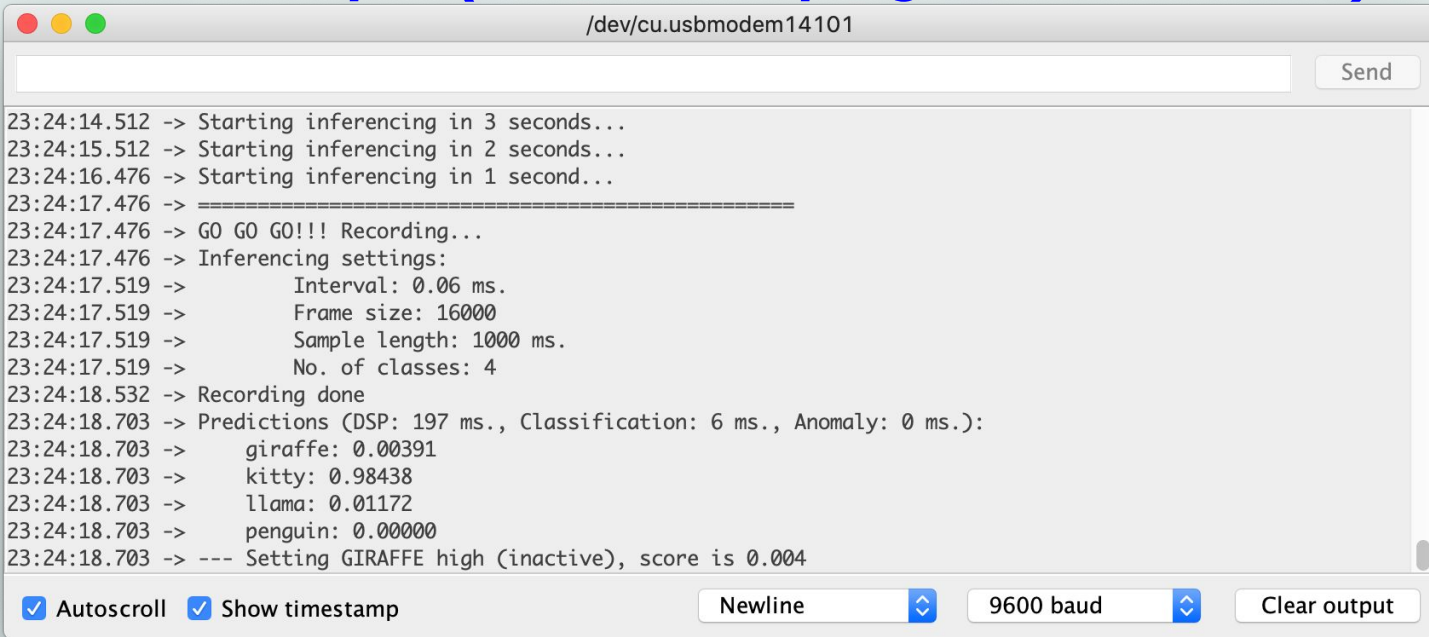
EI_IMPULSE_ERROR r = run_classifier(&signal, &result, debug_nn);
if (r != EI_IMPULSE_OK) {
    ei_printf("ERR: Failed to run classifier (%d)\n", r);
    return;
}

// print the predictions
ei_printf("Predictions ");
ei_printf("DSP: %d ms., Classification: %d ms., Anomaly: %d ms.",
    result.timing.dsp, result.timing.classification, result.timing.anomaly);
ei_printf("\n");
for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
    ei_printf("  %s: %.5f\n", result.classification[ix].label, result.classification[ix].value);
}
#if EI_CLASSIFIER_HAS_ANOMALY == 1
ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif

// ghills
if (result.classification[0].value > 0.99) {
    digitalWrite(2, LOW);
    ei_printf("--- Setting GIRAFFE low (active), score is %.3f\n", result.classification[0].value);
} else {
    digitalWrite(2, HIGH);
    ei_printf("--- Setting GIRAFFE high (inactive), score is %.3f\n", result.classification[0].value);
}
}
```

Step 2: Run it on Arduino

Status Output (for detailed program information)

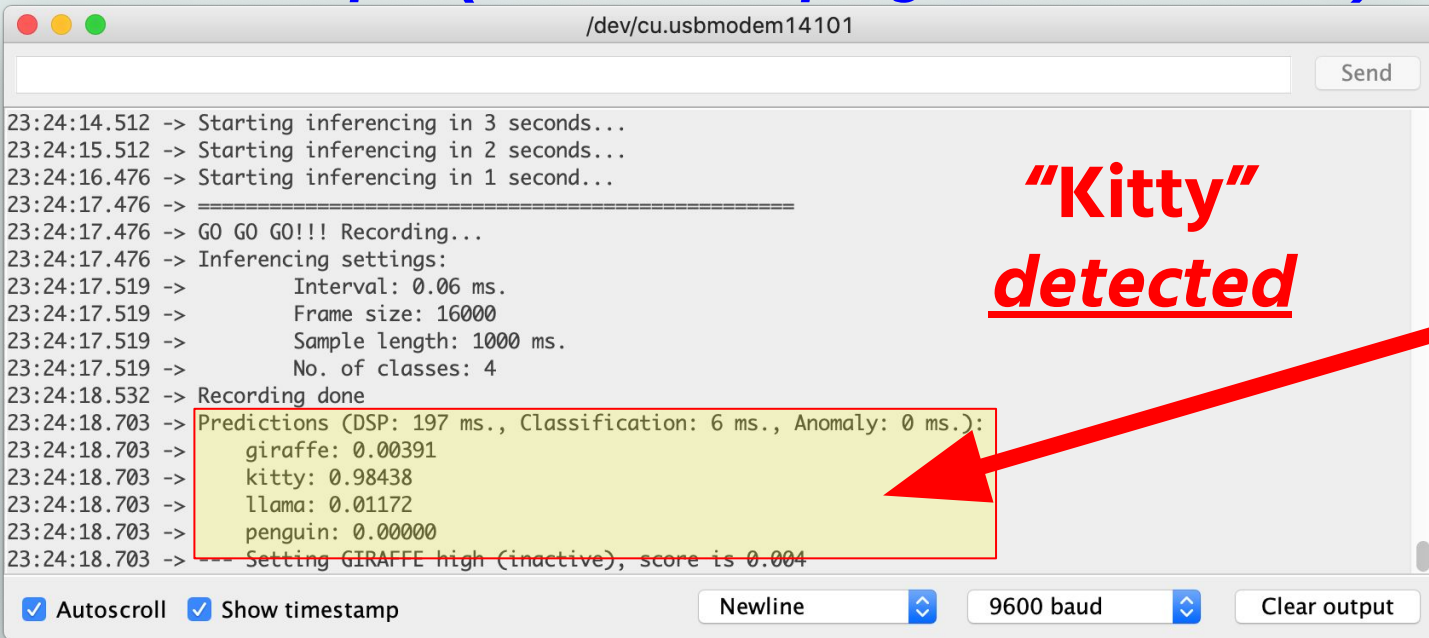


```
/dev/cu.usbmodem14101
23:24:14.512 -> Starting inferencing in 3 seconds...
23:24:15.512 -> Starting inferencing in 2 seconds...
23:24:16.476 -> Starting inferencing in 1 second...
23:24:17.476 -> =====
23:24:17.476 -> GO GO GO!!! Recording...
23:24:17.476 -> Inferencing settings:
23:24:17.519 ->     Interval: 0.06 ms.
23:24:17.519 ->     Frame size: 16000
23:24:17.519 ->     Sample length: 1000 ms.
23:24:17.519 ->     No. of classes: 4
23:24:18.532 -> Recording done
23:24:18.703 -> Predictions (DSP: 197 ms., Classification: 6 ms., Anomaly: 0 ms.):
23:24:18.703 ->     giraffe: 0.00391
23:24:18.703 ->     kitty: 0.98438
23:24:18.703 ->     llama: 0.01172
23:24:18.703 ->     penguin: 0.00000
23:24:18.703 -> --- Setting GIRAFFE high (inactive), score is 0.004

 Autoscroll  Show timestamp
Newline 9600 baud Clear output
```


Step 2: Run it on Arduino

Status Output (for detailed program information)



```
23:24:14.512 -> Starting inferencing in 3 seconds...
23:24:15.512 -> Starting inferencing in 2 seconds...
23:24:16.476 -> Starting inferencing in 1 second...
23:24:17.476 -> =====
23:24:17.476 -> GO GO GO!!! Recording...
23:24:17.476 -> Inferencing settings:
23:24:17.519 ->     Interval: 0.06 ms.
23:24:17.519 ->     Frame size: 16000
23:24:17.519 ->     Sample length: 1000 ms.
23:24:17.519 ->     No. of classes: 4
23:24:18.532 -> Recording done
23:24:18.703 -> Predictions (DSP: 197 ms., Classification: 6 ms., Anomaly: 0 ms.):
23:24:18.703 ->     giraffe: 0.00391
23:24:18.703 ->     kitty: 0.98438
23:24:18.703 ->     llama: 0.01172
23:24:18.703 ->     penguin: 0.00000
23:24:18.703 -> --- Setting GIRAFFE high (inactive), score is 0.004
```

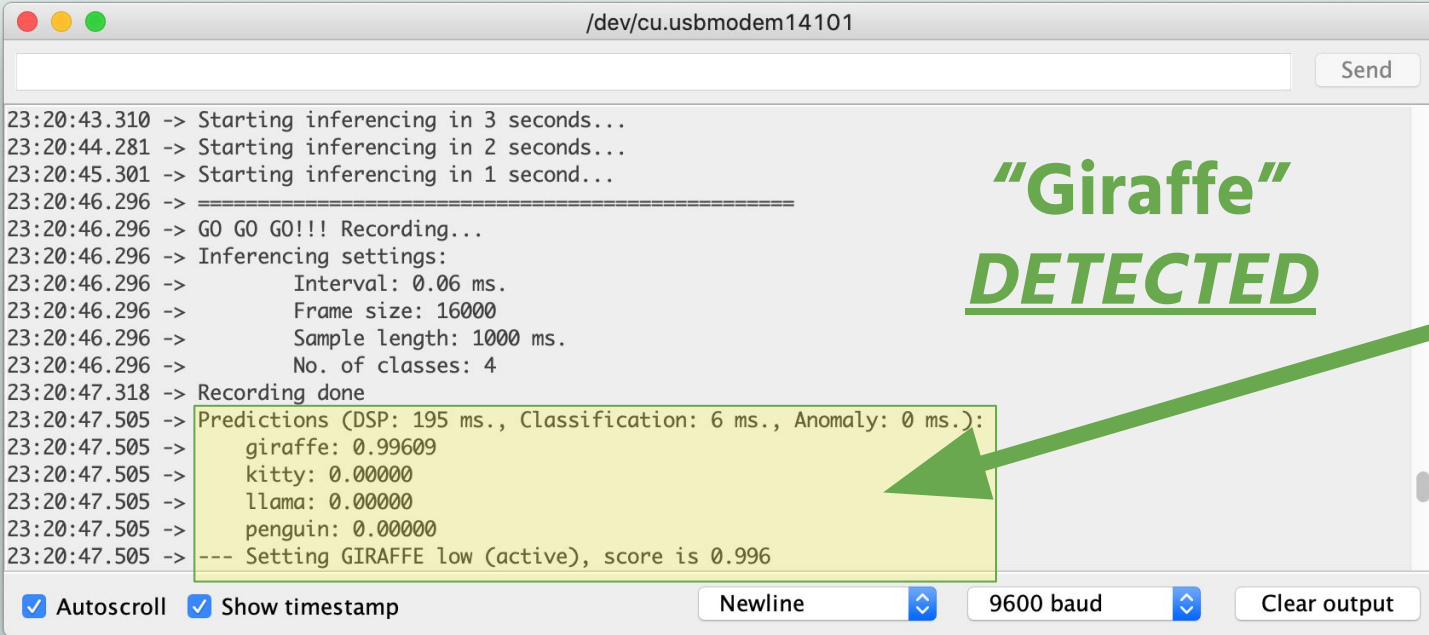
**"Kitty"
detected**

Autoscroll Show timestamp

Newline 9600 baud Clear output

Step 2: Run it on Arduino

Status Output (for detailed program information)



```
23:20:43.310 -> Starting inferencing in 3 seconds...
23:20:44.281 -> Starting inferencing in 2 seconds...
23:20:45.301 -> Starting inferencing in 1 second...
23:20:46.296 -> =====
23:20:46.296 -> GO GO GO!!! Recording...
23:20:46.296 -> Inferencing settings:
23:20:46.296 ->     Interval: 0.06 ms.
23:20:46.296 ->     Frame size: 16000
23:20:46.296 ->     Sample length: 1000 ms.
23:20:46.296 ->     No. of classes: 4
23:20:47.318 -> Recording done
23:20:47.505 -> Predictions (DSP: 195 ms., Classification: 6 ms., Anomaly: 0 ms.):
23:20:47.505 ->     giraffe: 0.99609
23:20:47.505 ->     kitty: 0.00000
23:20:47.505 ->     llama: 0.00000
23:20:47.505 ->     penguin: 0.00000
23:20:47.505 -> --- Setting GIRAFFE low (active), score is 0.996
```

**"Giraffe"
DETECTED**

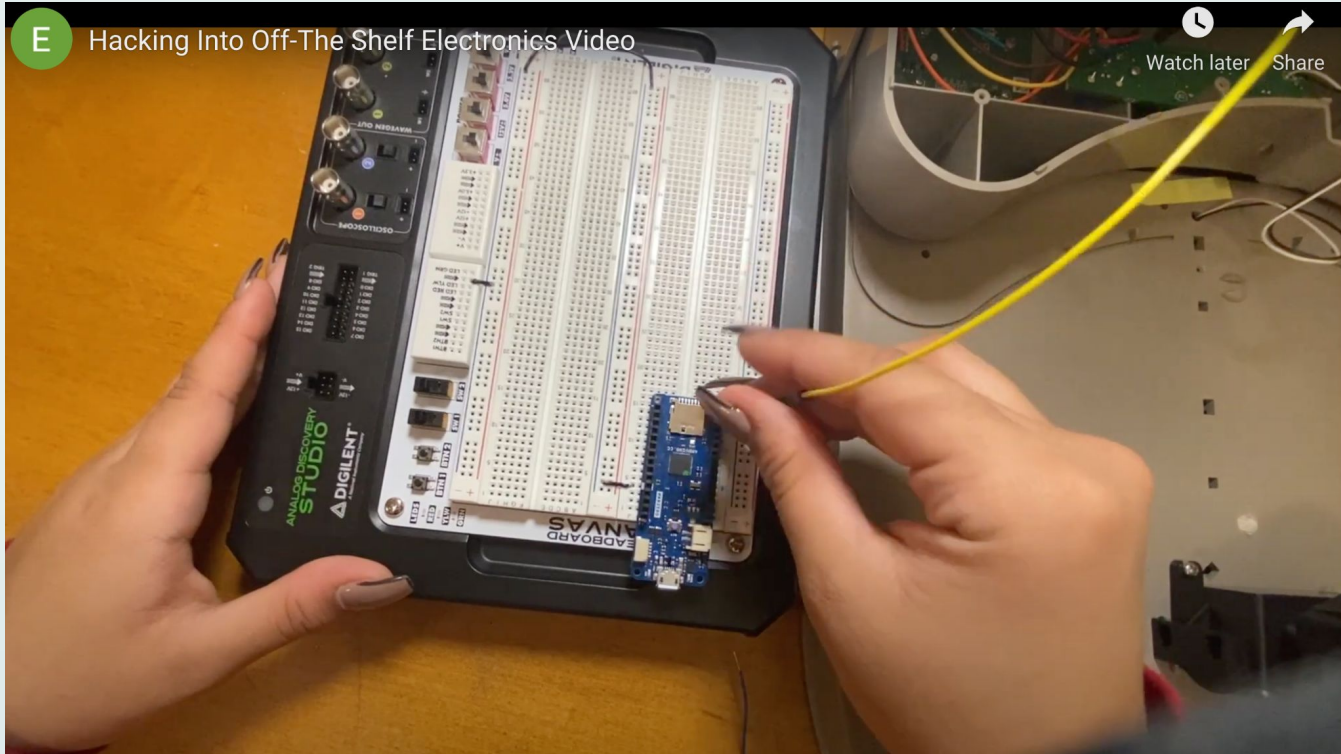
Autoscroll Show timestamp

Newline 9600 baud Clear output

Step 3: Connect it to Pet Feeder

E Hacking Into Off-The Shelf Electronics Video

Watch later Share



LIVE DEMO

featuring:

Chuckles

&

Cakmak-boy



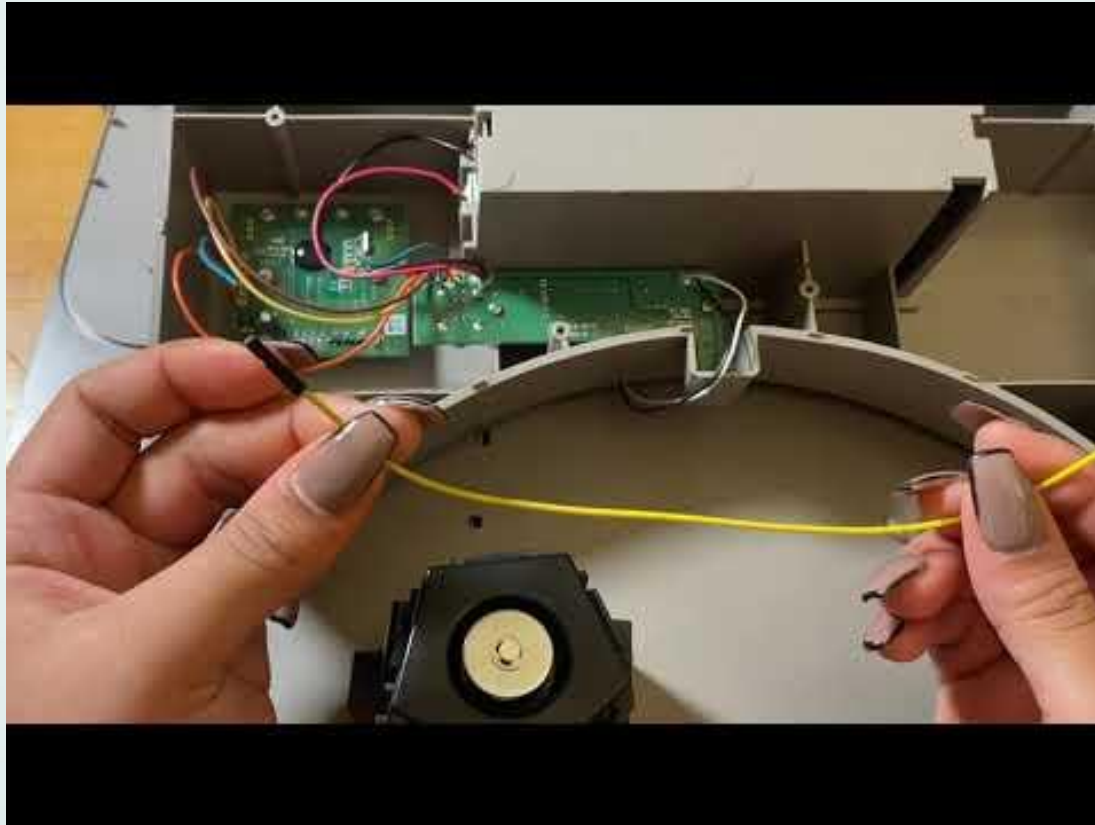
VIDEO WALKTHROUGH

starring:

Ezenia Diaz-Lembert



Video Walkthrough: Pet Feeder + Arduino



Questions ?

Step 1:

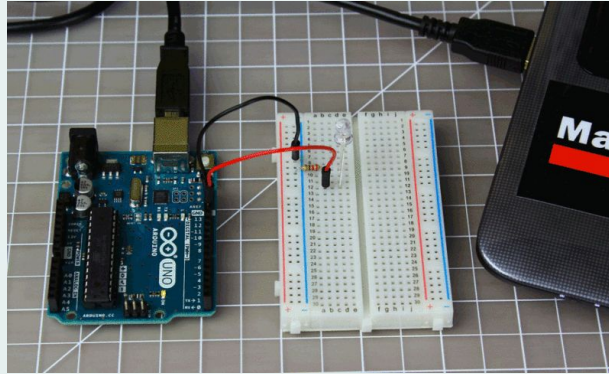
Build TinyML model



EDGE IMPULSE

Step 2:

Run it on Hardware



Arduino BLE Nano 33

Step 3:

Feed my cats



Pet feeder

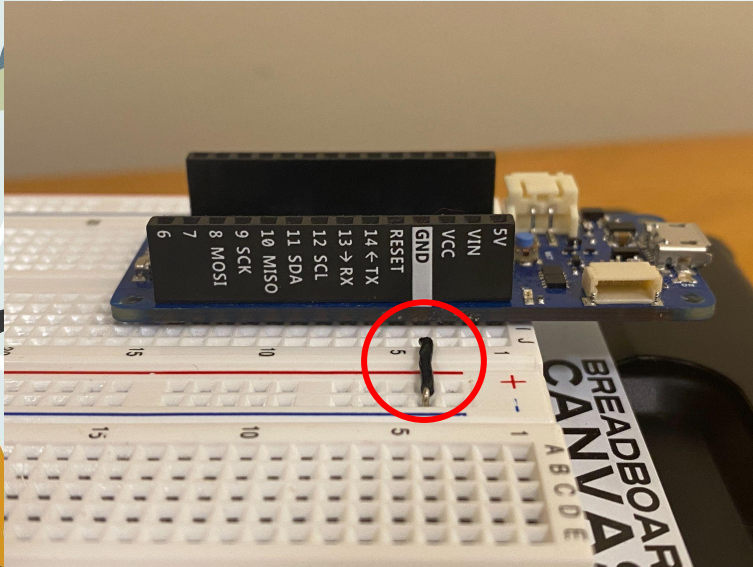
The image features a light blue background with a central white rectangular box containing the text "Thank You!". The text is rendered in a bold, orange, sans-serif font. Surrounding the text box are various abstract geometric elements: circles in shades of teal, orange, black, and white, and lines of varying colors (teal, orange, black) that connect some of the circles, creating a network-like structure. The overall aesthetic is clean and modern.

Back-up slides

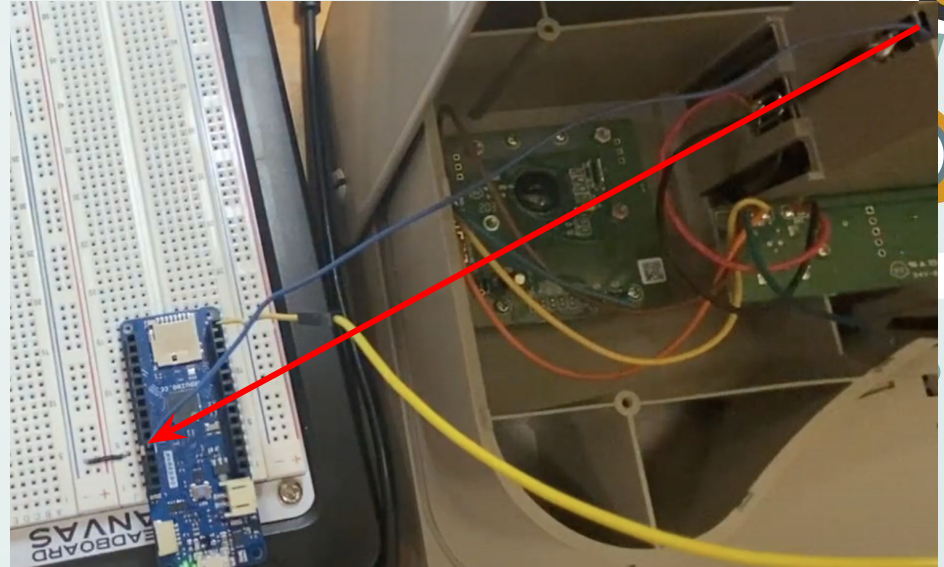


Debugging!

CHECK: Did you connect **both** the Arduino and feeder to GND?



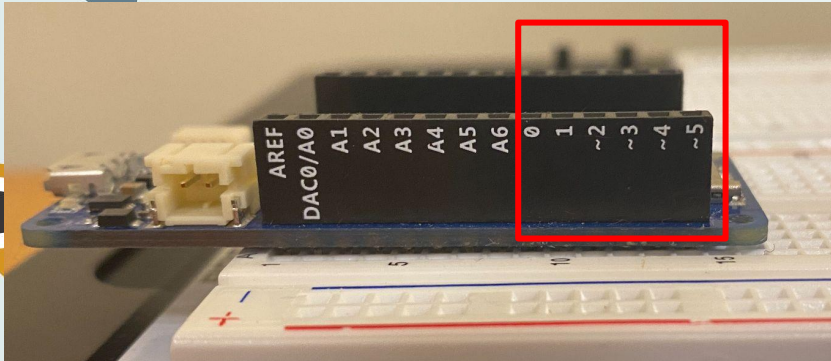
^Arduino Connection to GND



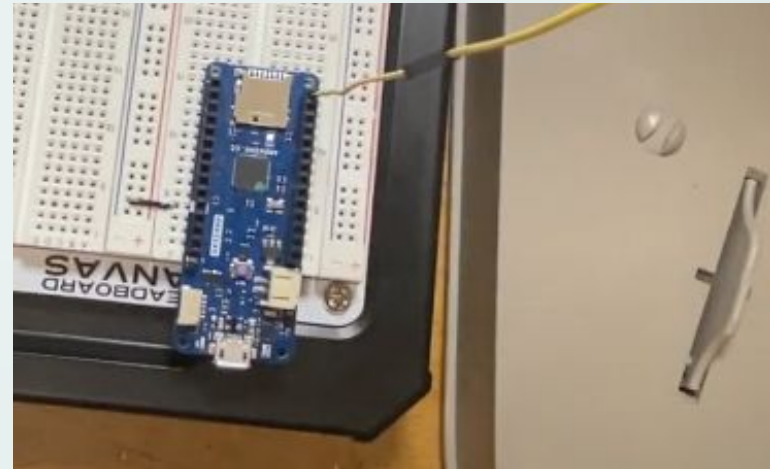
^Feeder Connection to GND (Blue Wire in Video)

Debugging!

CHECK: Did you connect the Feeder's PWR to the Arduino's? Make sure it matches with your digital pin in your Arduino Code!



^Digital Pins



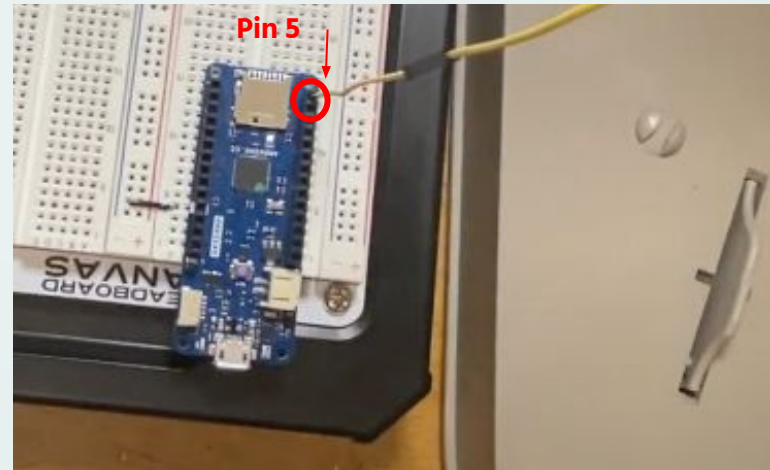
^Feeder Connection to PWR (Yellow Wire in Video)

Debugging!

CHECK: Did you connect the Feeder's PWR to the Arduino's?
Make sure it matches with your digital pin in your Arduino Code!

```
sketch_apr13a | Arduino 1.8.13  
sketch_apr13a $  
void setup() {  
  // put your setup code here, to run once:  
  pinMode(5, OUTPUT); // sets digital pin 5 as the output  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
  digitalWrite(5, HIGH); // sets digital pin 5 to HIGH  
  delay(2000); // sets a delay of 2 seconds  
  digitalWrite(5, LOW); // sets digital pin 5 to LOW, off. Should call  
  delay(2000); // sets delay of 2 seconds  
  
  // code will loop through!  
}
```

^Arduino IDE pin assignment



^Feeder Connection to PWR (Yellow Wire in Video)

Debugging!

CHECK: 3.3 V check

- If you're having trouble with the voltage drop/ it's not working, it'd be a great idea to check the Voltage we're getting from the Arduino
- When set to HIGH, we should read 3.3V.
- From our code, the Voltage being read should go from 3.3V to 0V, switching every ~2 seconds
- To measure voltage, we can use a Voltmeter/ Multimeter as shown!





Thank You!

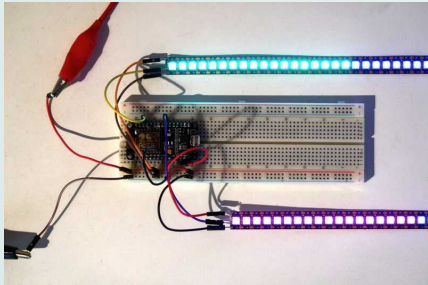


Hacking off Into Off-The Shelf Electronics

Project: Automatic Pet Feeder

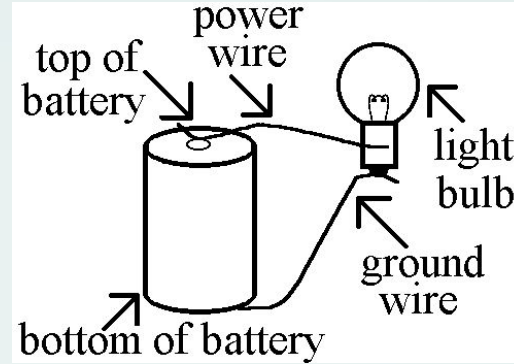
Applications

- Much of the same ML and Arduino concepts you've been learning about are seen in electronics all around us!
- What are some electronic devices you can think of? How do you think they work?



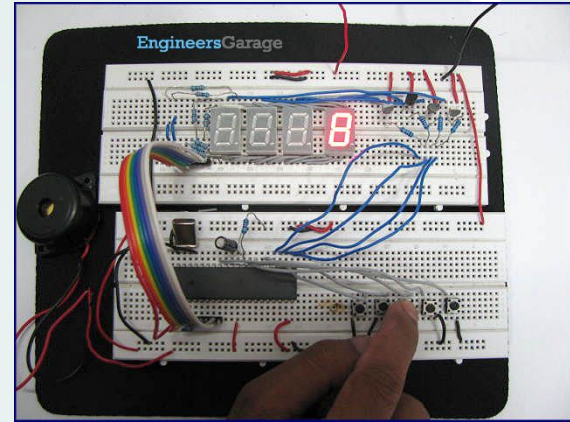
Applications

- In order for devices to perform tasks, we need some energy or power to help them do so!
- Electronic devices use some power source (require batteries, an outlet, solar panel, etc.) that it can use to perform certain tasks (ex: ring a digital alarm clock at a set time by the user).
- While electronic devices connect to power, they Also connect to Ground (GND). You can see the GND pin on your Arduino, a board which can also take a power source (ex: VCC pin) of 3.3 Volts
- Using the Arduino Integrated Development Environment, we can add some programming to control devices around us.



Applications

- Since many electronic devices are guided by the same main components, that means that we can also look “under the hood” at the wires and boards that come together to make these devices work as they should.
- This means that just as how many electronic devices around us have come together, we can take them apart to program and modify! A clear example would be controlling our input power.
- Ever wanted to change how some device at home worked? For example, do you wish a certain device could move faster, have a sensor, etc. ? You can get as creative as possible and take what you already have to the next level.





Today, we're going to take
apart and control an
Electronic Device!

The Automatic Pet Feeder



In particular, we're going to be "hacking" into an automatic pet feeder!

This feeder is made to rotate at set times to feed pets! So... we know a couple things:

- Some power is driving the feeding container to move (motor)
- Some programming is being used to control this device!

This will be incredibly useful for our hacking!

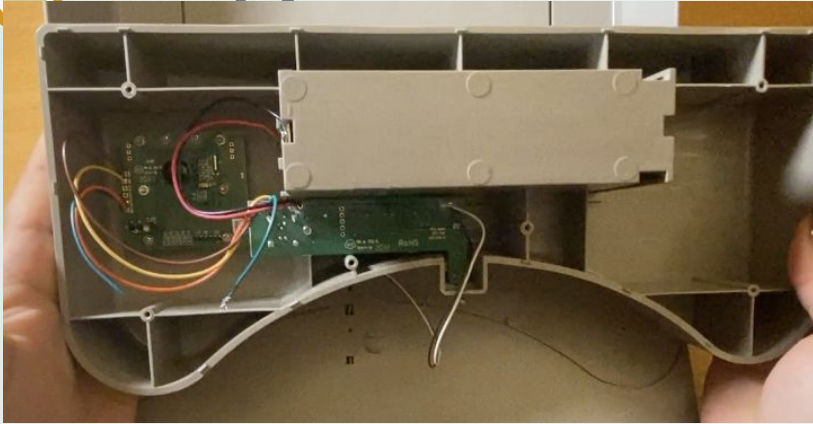
The Automatic Pet Feeder



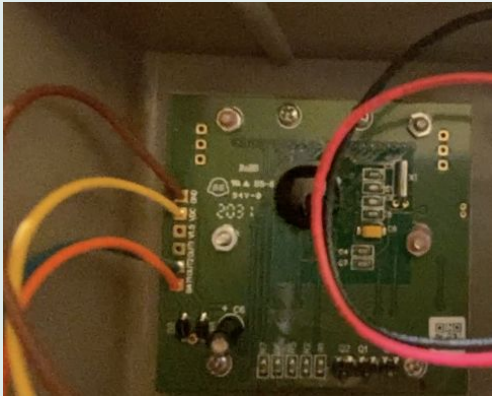
Once you have opened the feeder, it should look like this!

- Here we can see the removable container (if you take it off, you can see the motor), and control system of the pet feeder
- But, how does the pet feeder know **when** to rotate ? How does it know how to rotate at all?

How Does it Work?



- After unscrewing the back of the pet feeder, you should see something like this beneath the control system of the feeder
- Consists of boards taking into account the digital signals and programming set by the user from the buttons, and the control of power to the motor



How Does it Work?



Power is directly related to voltage and current. For the pet feeder, a voltage drop of 1.5 V causes the motor to move!

Wires:

- Black and White → Motor
- Yellow, Brown, Green → Digital Control Board
- VVC, GND → Power for digital chip
- Motor needs a higher voltage, control signal given by OUTZ.
- For us to control the signal for motor movement, we should cut OUTZ (Blue).

$$P = IV$$

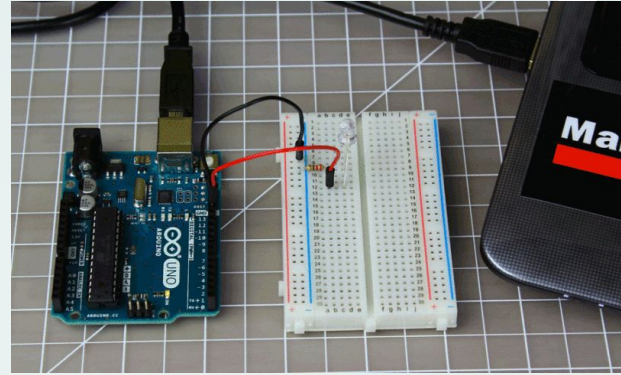
P = power (watts, W)

I = current (amperes, A)

V = voltage (volts, V)

How Can We Control Power?

- We can actually use our Arduinos and the Arduino IDE to control the power!
- This way, we can code for the voltage drop we need for the motor to move.



```
sketch_dec07a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_dec07a
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
Arduino/Genuino Uno on COM3
```



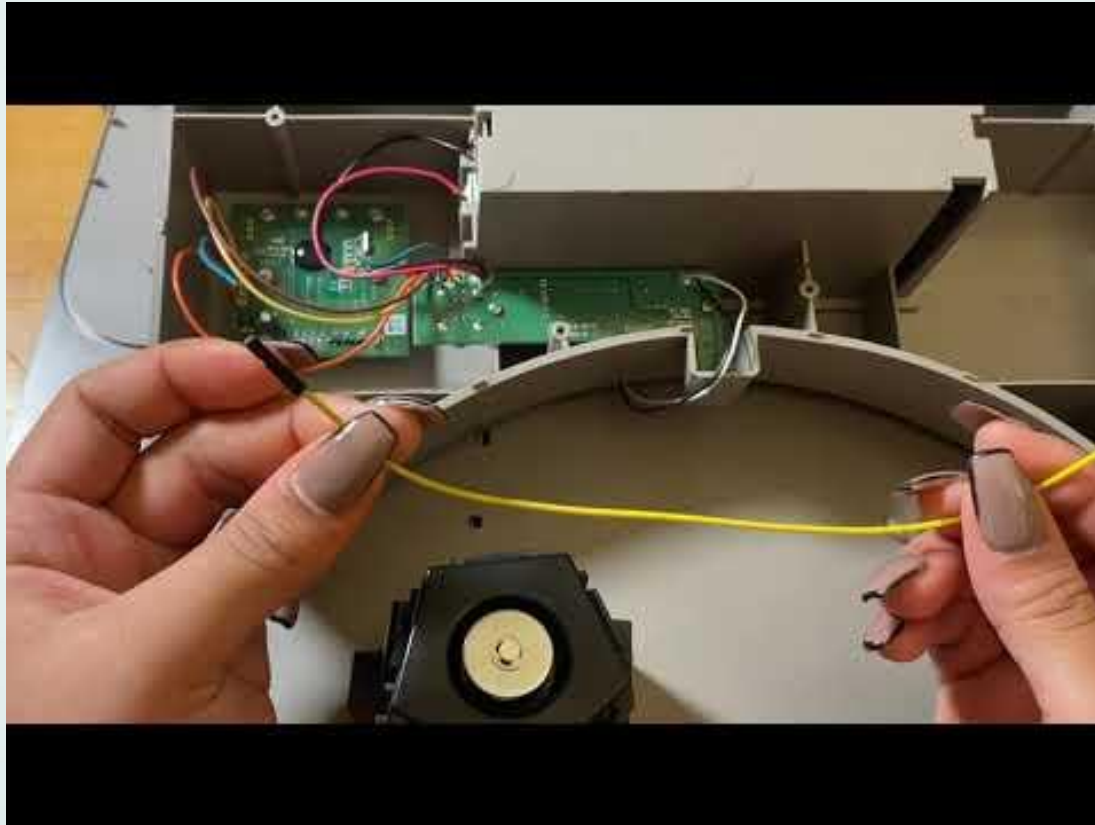
Connecting to an Arduino

```
void setup() {  
  pinMode(5, OUTPUT); // digital pin 5 is the output  
}  
  
void loop() {  
  digitalWrite(5, HIGH); // digital pin 5 on  
  delay(2000);           // waits for two seconds  
  digitalWrite(5, LOW); // digital pin 5 off  
  delay(1000);          // waits for two seconds  
}
```

- This code goes back and forth, setting voltage from HIGH to LOW (0V, GND... this leads to the motor moving!)

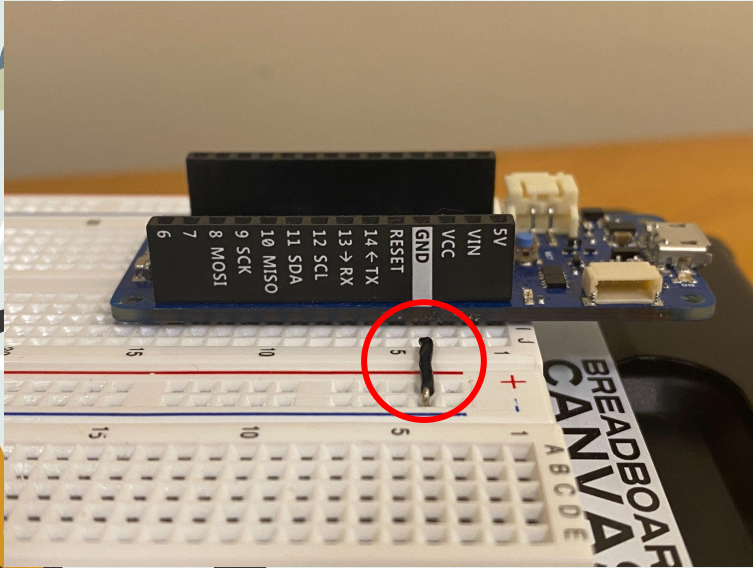
- We need to program for this Voltage drop
- In other words, when the Voltage goes from HIGH to LOW
- We can set the pins on the Arduino from HIGH to LOW on the Arduino Integrated Development Environment (IDE)
- We can set the voltage from an output digital pin (in this example I use pin 5)

Video Walkthrough

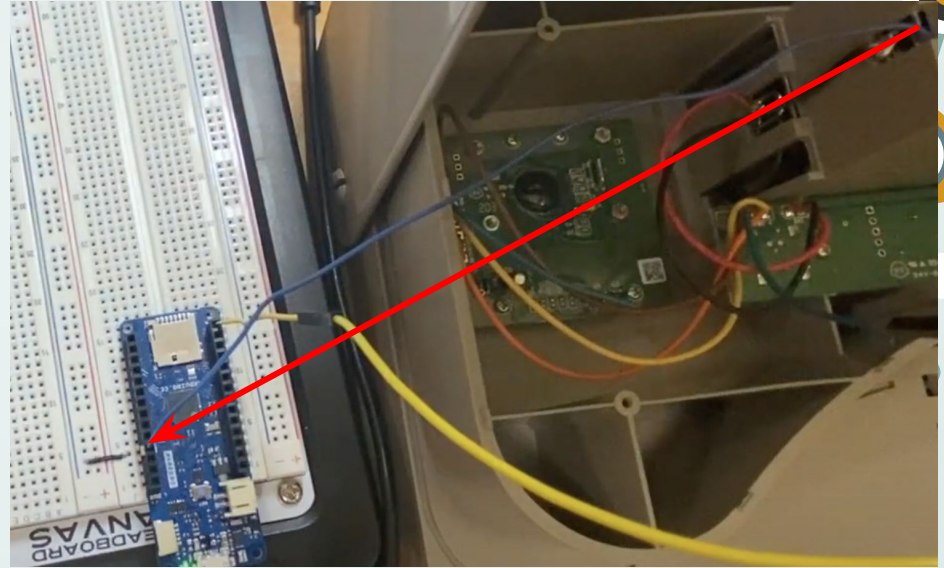


Debugging!

CHECK: Did you connect **both** the Arduino and feeder to GND?



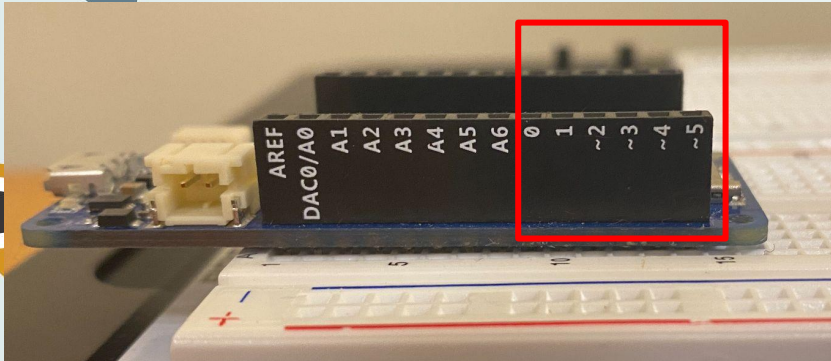
^Arduino Connection to GND



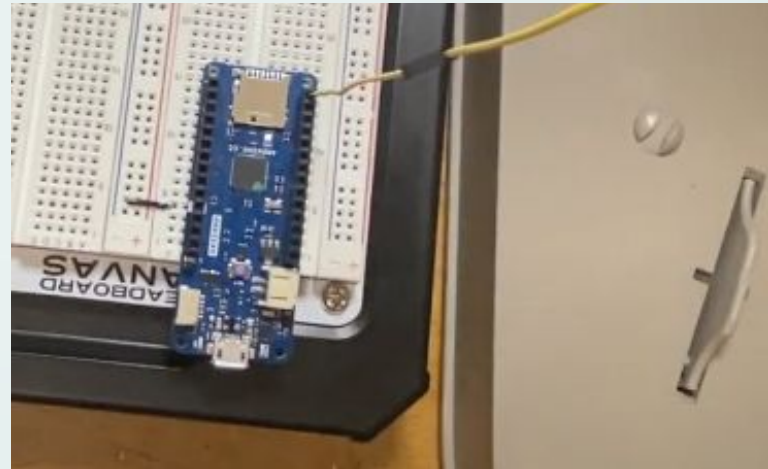
^Feeder Connection to GND (Blue Wire in Video)

Debugging!

CHECK: Did you connect the Feeder's PWR to the Arduino's? Make sure it matches with your digital pin in your Arduino Code!



^Digital Pins



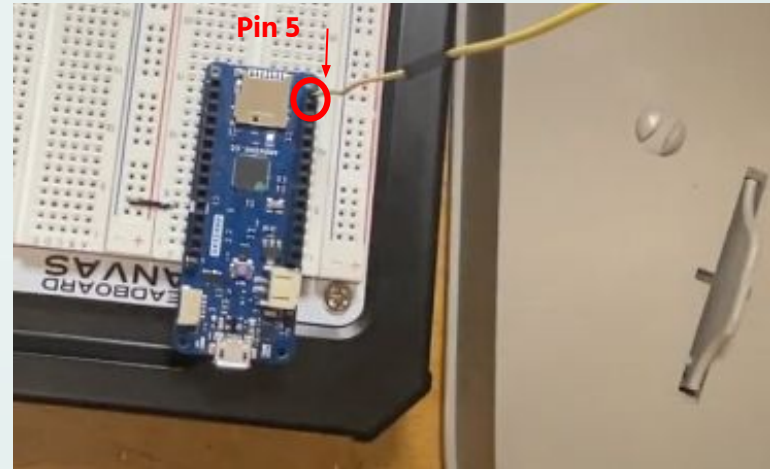
^Feeder Connection to PWR (Yellow Wire in Video)

Debugging!

CHECK: Did you connect the Feeder's PWR to the Arduino's?
Make sure it matches with your digital pin in your Arduino Code!

```
sketch_apr13a | Arduino 1.8.13  
sketch_apr13a $  
void setup() {  
  // put your setup code here, to run once:  
  pinMode(5, OUTPUT); // sets digital pin 5 as the output  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
  digitalWrite(5, HIGH); // sets digital pin 5 to HIGH  
  delay(2000); // sets a delay of 2 seconds  
  digitalWrite(5, LOW); // sets digital pin 5 to LOW, off. Should call  
  delay(2000); // sets delay of 2 seconds  
  
  // code will loop through!  
}
```

^Arduino IDE pin assignment



^Feeder Connection to PWR (Yellow Wire in Video)

Debugging!

CHECK: 3.3 V check

- If you're having trouble with the voltage drop/ it's not working, it'd be a great idea to check the Voltage we're getting from the Arduino
- When set to HIGH, we should read 3.3V.
- From our code, the Voltage being read should go from 3.3V to 0V, switching every ~2 seconds
- To measure voltage, we can use a Voltmeter/ Multimeter as shown!



The image features a light blue background with a central white rectangular box containing the text "Thank You!". The text is rendered in a bold, orange, sans-serif font. Surrounding the text box are various abstract elements: thin white lines connecting circular nodes in shades of teal, orange, and black; scattered solid circles in white, black, teal, and orange; and larger, teardrop-shaped elements in teal and orange. The overall aesthetic is clean, modern, and professional.

Thank You!