



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

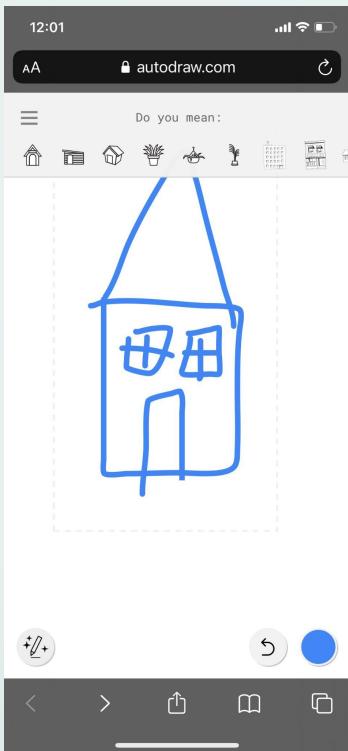
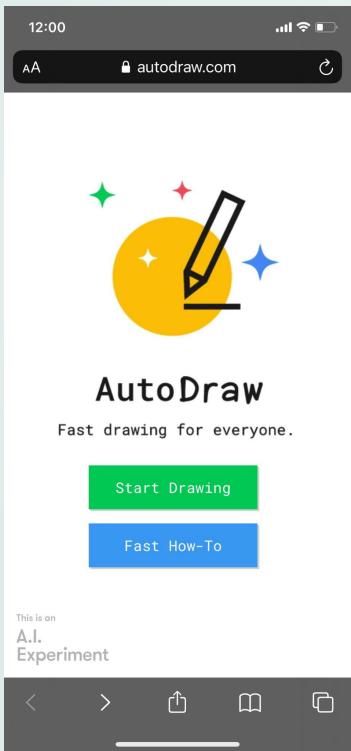
Artificial
Intelligence

Classification

Vision

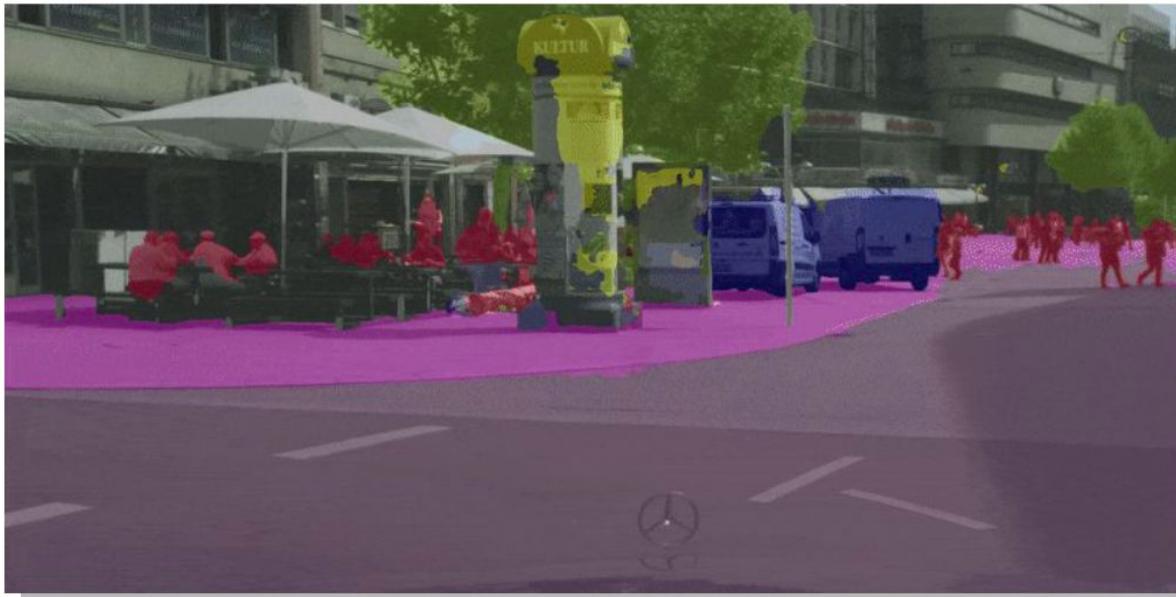
Audio

So Far



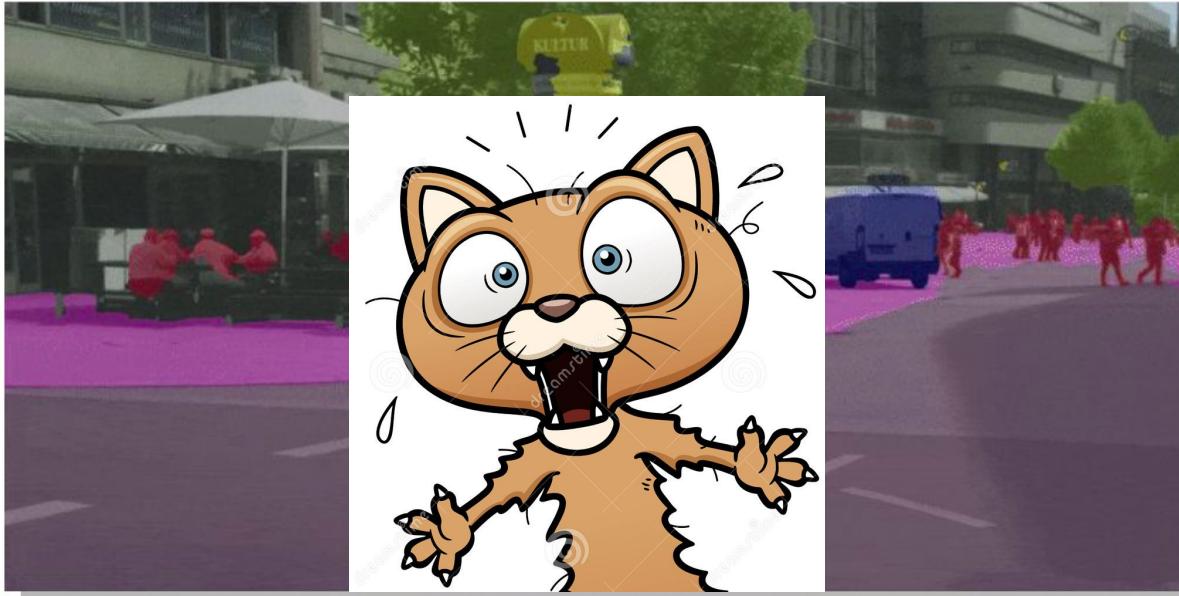
So Far

Segmentation



So Far

"Cat" detected



Then What ?

Cat detected...



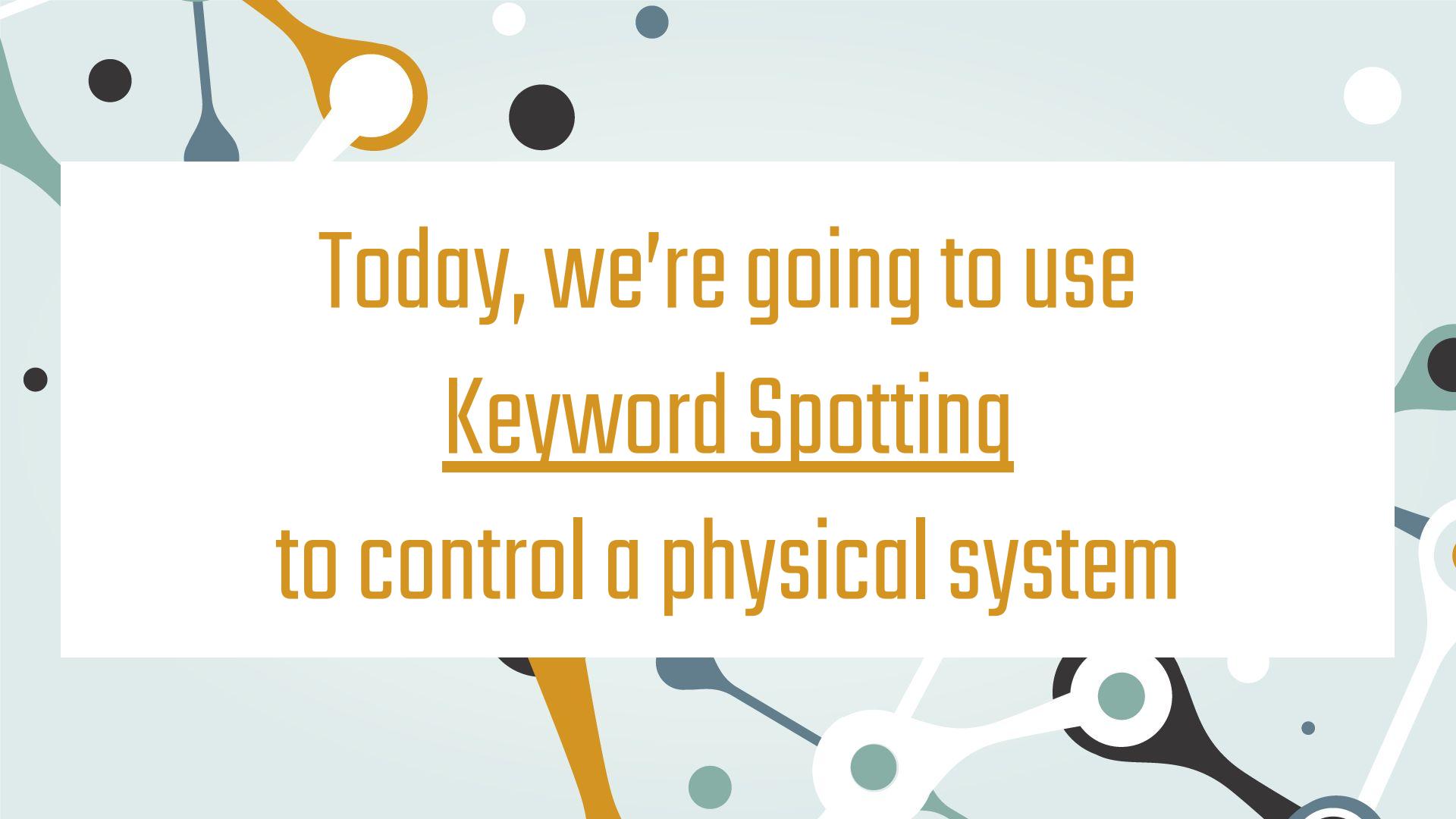
Then What ?

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!

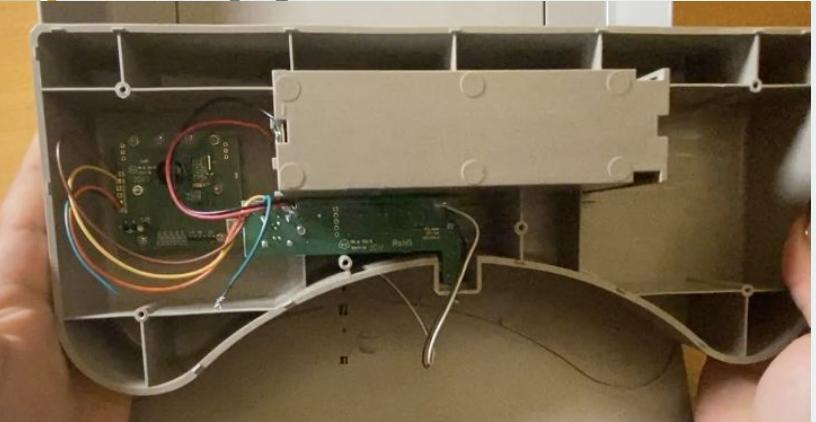
The Automatic Pet Feeder



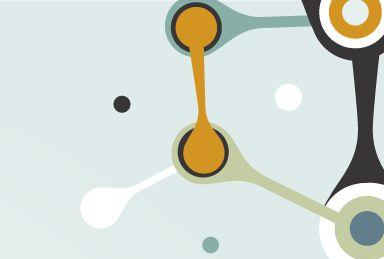
Once you have opened the feeder, it should look like this!

- 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

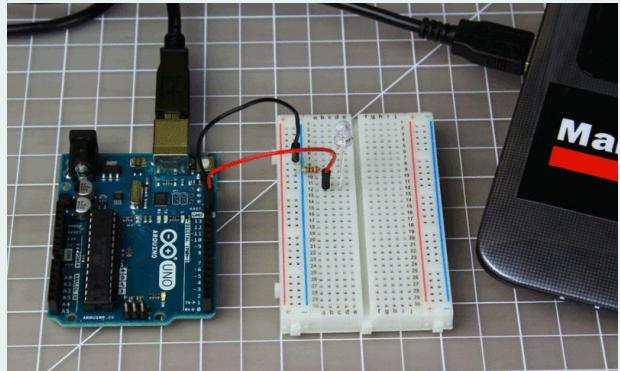


Today

Step 1:
Build TinyML model



Step 2:
Run it on Hardware



Step 3:
Feed my cats



 **EDGE IMPULSE**

Arduino BLE Nano 33

Pet feeder

Step 1: Build TinyML Model

The screenshot shows the Edge Impulse web interface for a project titled "Gage Hills / cat".

Project info | Keys | Export

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

EDGE 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

EDGE IMPULSE

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

DATA ACQUISITION (CAT)

Training data Test data

Did you know? You can capture data from any device or development board, or upload your existing datasets - [Show options](#)

DATA COLLECT... 1m 0s

LABELS 4

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	⋮

Record new data [Connect using WebUSB](#)

No devices connected to the remote management API.

RAW DATA
Click on a sample to load...

Step 1: Build TinyML Model



Step 1: Build TinyML Model

 **EDGE IMPULSE**

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

CREATE IMPULSE (CAT)

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

Window increase

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

Step 1: Build TinyML Model

EDGE IMPULSE

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

MFCC (CAT)

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](#)

Gage Hills

Feature explorer (60 samples)

X Axis Y Axis Z Axis

Visualization Visualization Visualization

giraffe kitty llama penguin

This figure is a 3D scatter plot titled 'Feature explorer (60 samples)'. It displays data points for four classes: giraffe (blue), kitty (orange), llama (green), and penguin (red). The plot is divided into three vertical layers labeled 'Visualization layer 3' at the top, 'Visualization layer 2' in the middle, and 'Visualization layer 1' at the bottom. The horizontal axes are labeled 'X Axis', 'Y Axis', and 'Z Axis', each with a dropdown menu set to 'Visualization'. The plot shows a complex, non-linear distribution of points, with each class forming distinct clusters across the three layers.

Step 1: Build TinyML Model

 **EDGE IMPULSE**

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

NN CLASSIFIER (CAT)

#1 ▾ Click to set a description for this version

Neural Network settings

Training settings

- Number of training cycles ⑦ 100
- Learning rate ⑦ 0.005
- Minimum confidence rating ⑦ 0.90

Audio training options

Data augmentation ⑦

Neural network architecture

Architecture presets ⑦

- 1D Convolutional (Default)
- 2D Convolutional

Training output

Model Model version: ⑦ Quantized (int8) ▾

Last training performance (validation set)

 ACCURACY	75.0%	 LOSS	0.54
---	--------------	---	-------------

Confusion matrix (validation set)

	GIRAFFE	KITTY	LLAMA	PENGUIN
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) ⑦

Step 2: Run it on Arduino

 **EDGE IMPULSE**

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

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

options.

Enable EON™ Compiler
Same accuracy, up to 50% less memory. Open source.

 **Built Arduino library**

Add this library through the Arduino IDE via:
Sketch > Include Library > Add .ZIP Library...

Examples can then be found under:
File > Examples > cat_inferencing

Build

Build output

Creating job... OK (ID: 1015061)

SDK...
SDK OK
.
K
updating headers...
updating headers OK

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

Step 2: Run it on Arduino

```
nano_ble33_sense_microphone | Arduino 1.8.15

nano_ble33_sense_microphone §

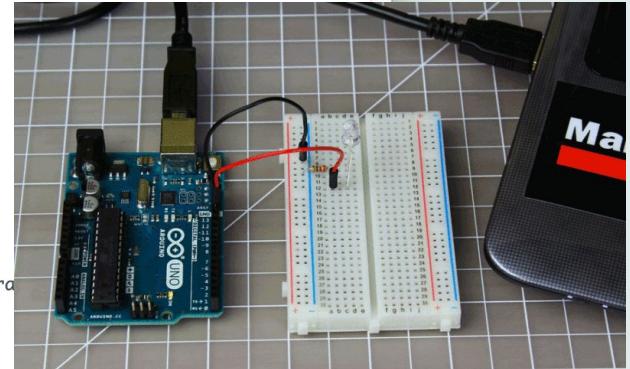
/*
 * Includes --
 #include <PDM.h>
 #include <cat_inferencing.h>

 /**
 * 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) / sizeof(
        ei_classifier_inferencing_category));
    ei_printf("\tSampling rate: %d Hz\n", EI_CLASSIFIER_RAW_SAMPLE_RATE);

    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

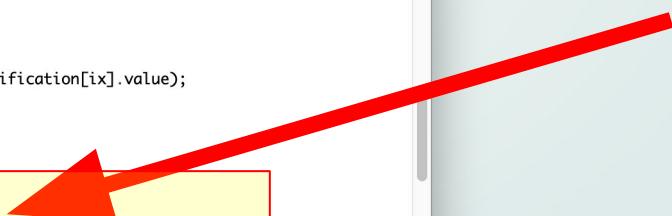
```
nano_ble33_sense_microphone | Arduino 1.8.15

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);
}
#endif
#if EI_CLASSIFIER_HAS_ANOMALY == 1
    ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif

// ghillis
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)

```
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

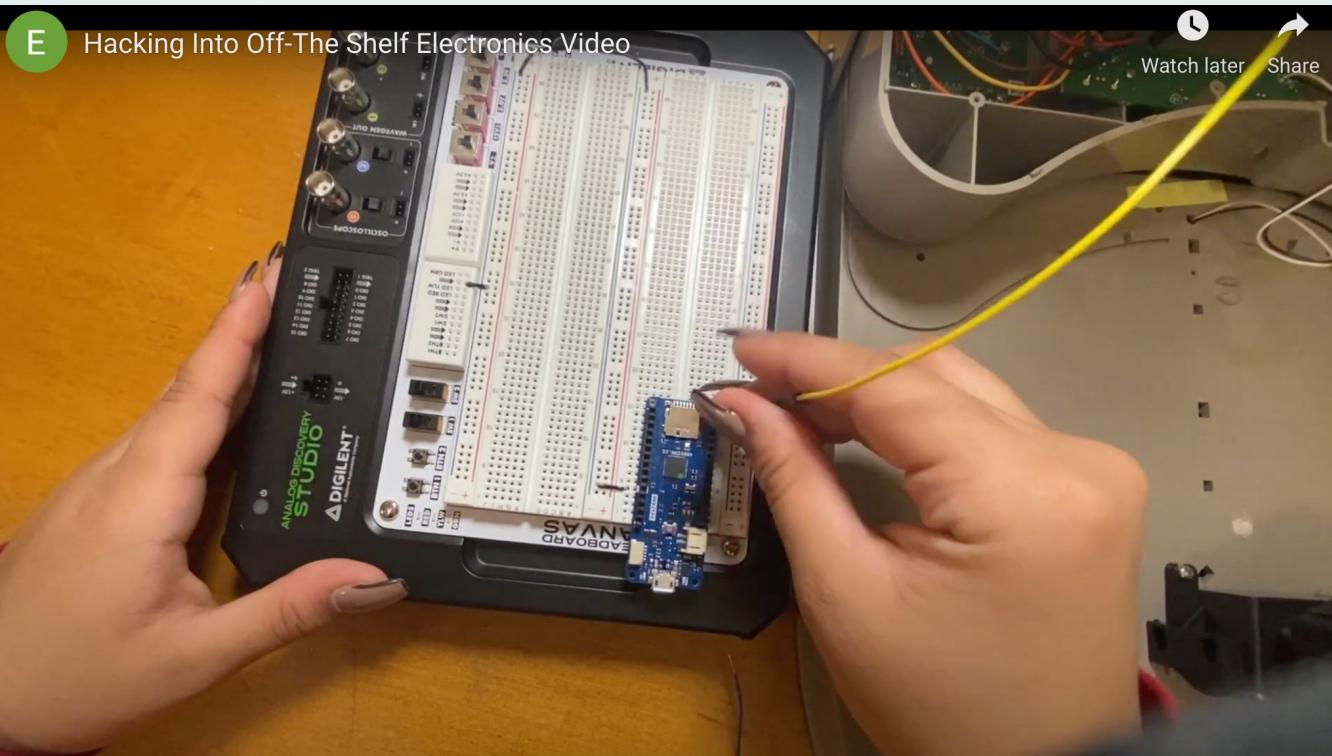
Status Output (for detailed program information)

```
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
```

Autoscroll Show timestamp Newline 9600 baud Clear output

“Giraffe”
DETECTED

Step 3: Connect it to Pet Feeder



LIVE DEMO

featuring:

Chuckles

&

Cakmak-boy



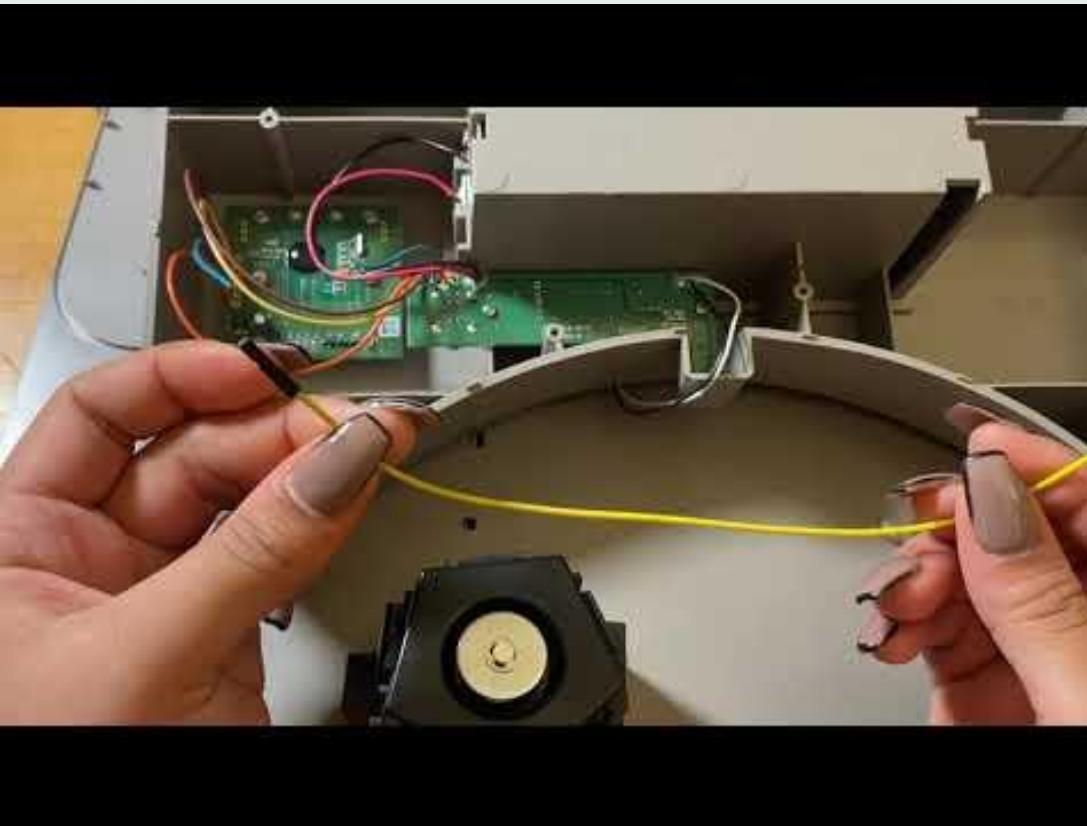
VIDEO WALKTHROUGH

starring:

Ezenia Diaz-Lembert



Video Walkthrough: Pet Feeder + Arduino

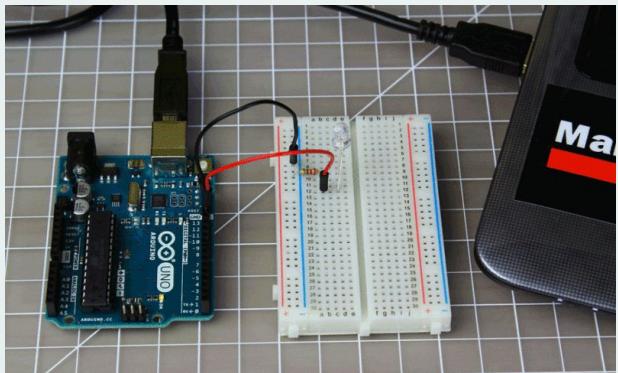


Questions ?

Step 1:
Build TinyML model



Step 2:
Run it on Hardware



Step 3:
Feed my cats



 **EDGE IMPULSE**

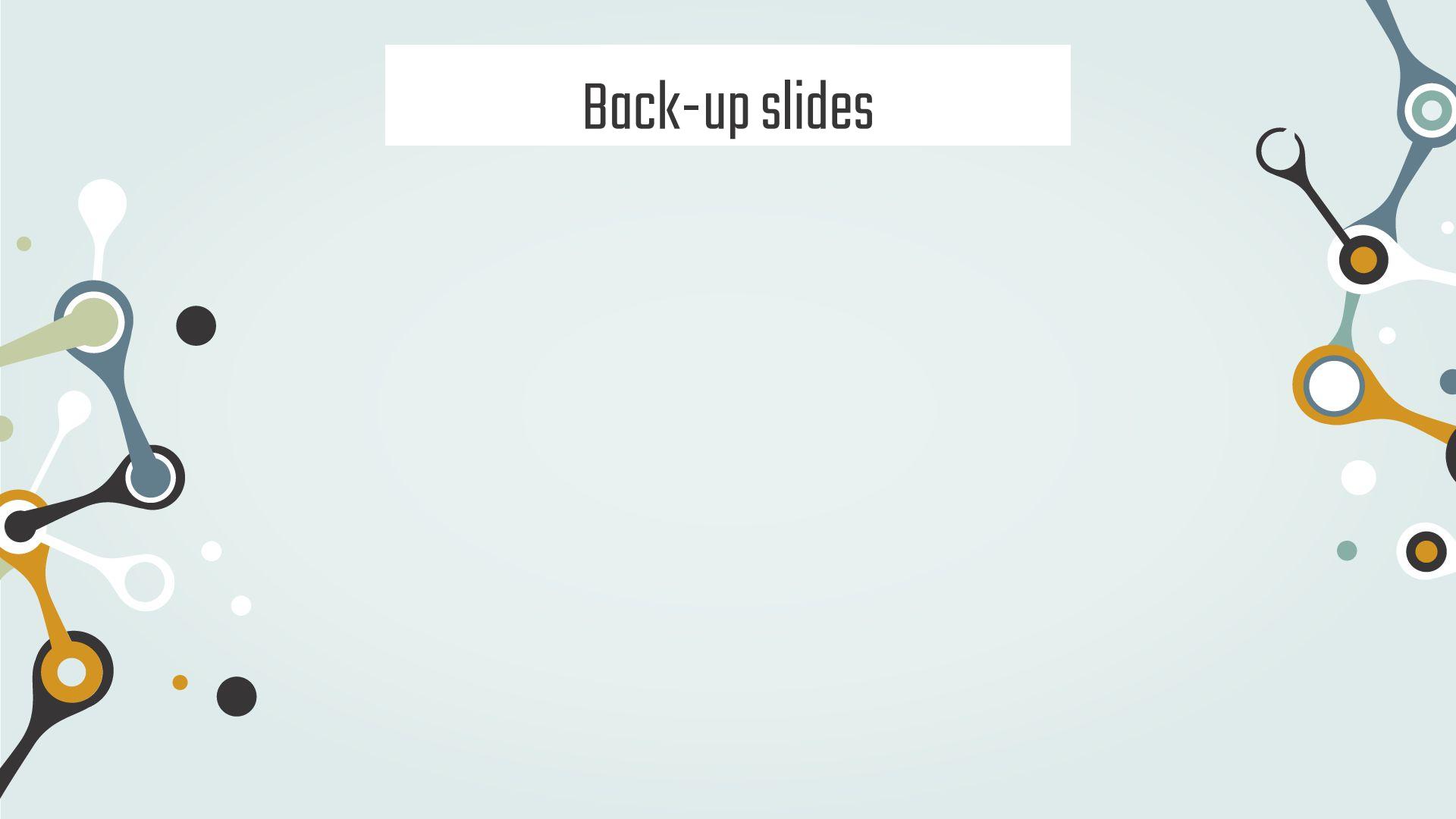
Arduino BLE Nano 33

Pet feeder



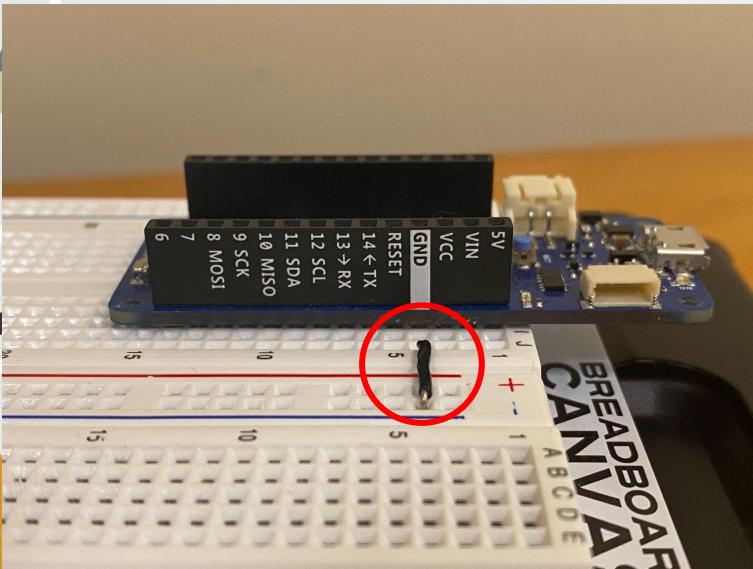
Thank You!

Back-up slides

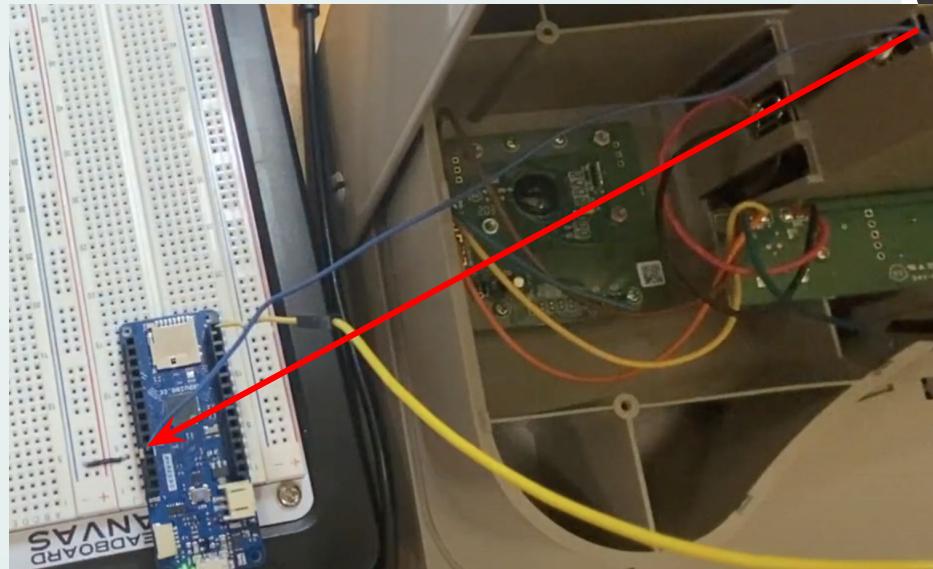


Debugging!

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



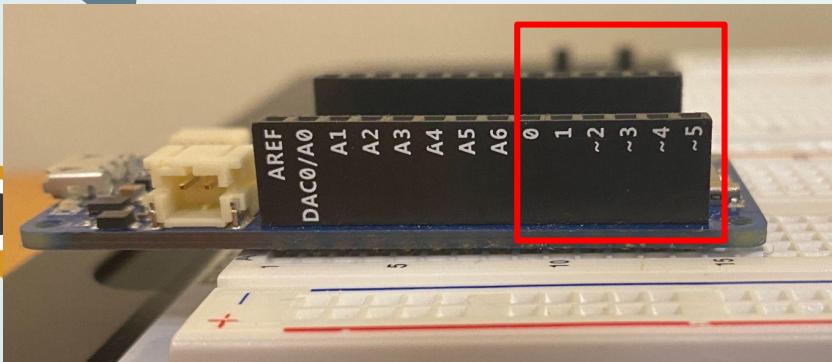
^Arduino Connection to GND



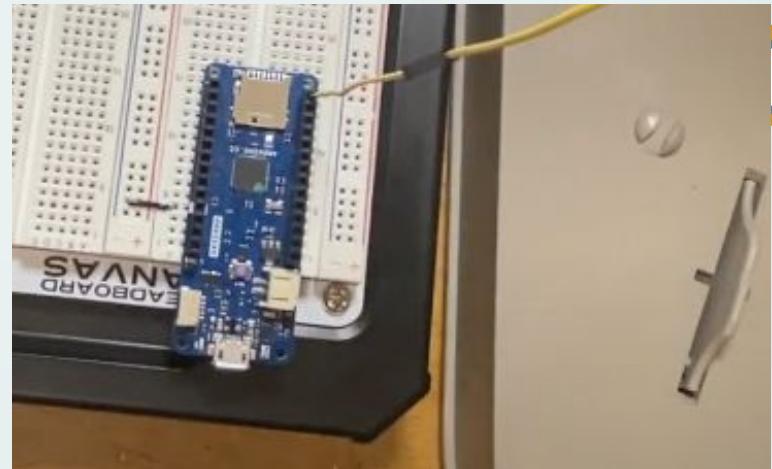
^Feeder Connection to GND (Blue Wire in Video)

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 §

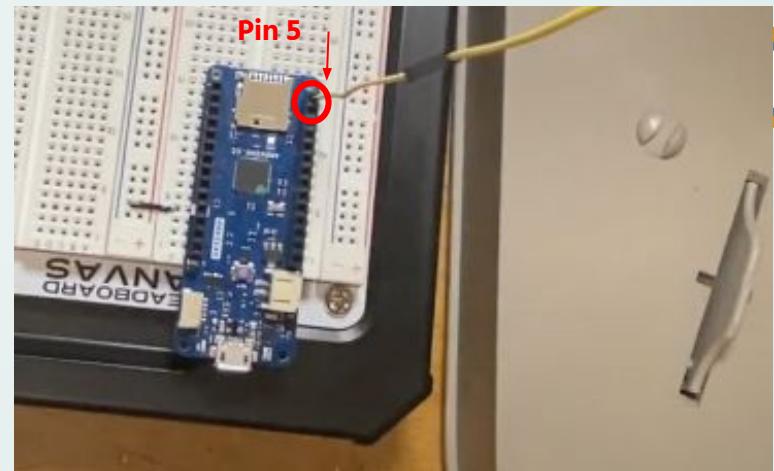
void setup() {
  // put your setup code here, to run once:
  pinMode(5, OUTPUT); // sets digital pin 5 as the output
}

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!



A white rectangular box containing the text "Thank You!" is centered on a light blue background. The background features a network of abstract, organic shapes in various colors including orange, teal, and grey, resembling stylized neurons or cells.

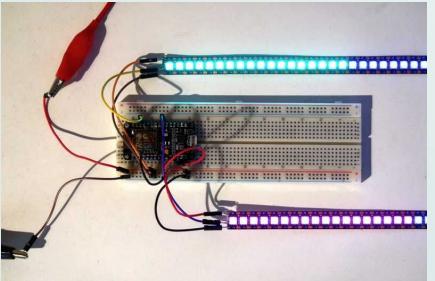
Thank You!

Hacking off Into Off-The Shelf Electronics

Project: Automatic Pet Feeder

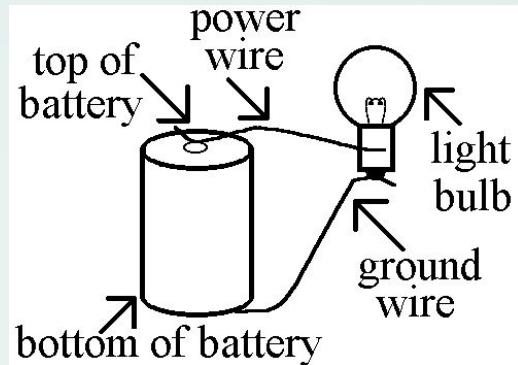
Applications

- 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?



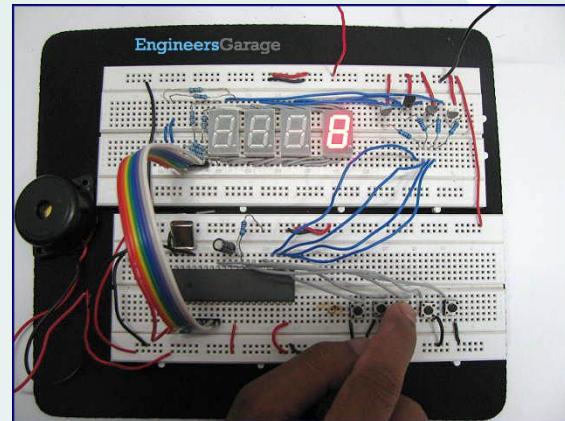
Applications

- 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!

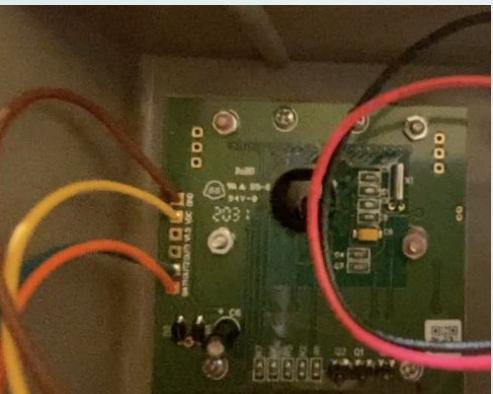
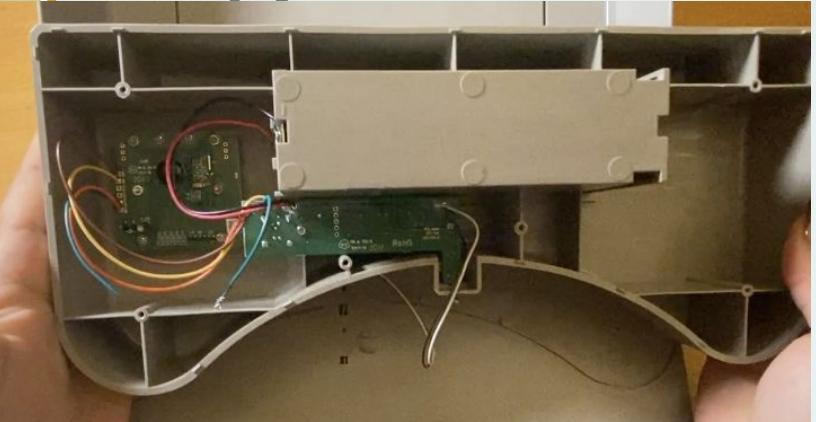
The Automatic Pet Feeder



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?



$$P = IV$$

P = power (watts, W)

I = current (amperes, A)

V = voltage (volts, V)

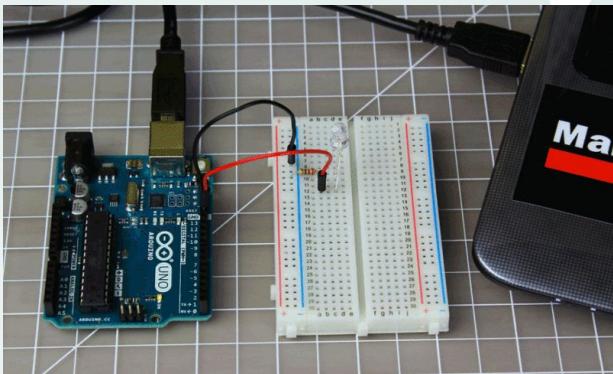
Power is directly related to voltage and current. For the pet feeder, a voltage drop of 1.5 V causes the motor to move!

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).

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:
}
```

2

Arduino/Genuino Uno on COM3

A screenshot of the Arduino IDE showing a simple sketch named "sketch_dec07a". The sketch contains two functions: "setup()" and "loop()". The "setup()" function is empty, and the "loop()" function is also empty. The Arduino board is set to "Arduino/Genuino Uno" and is connected to "COM3".



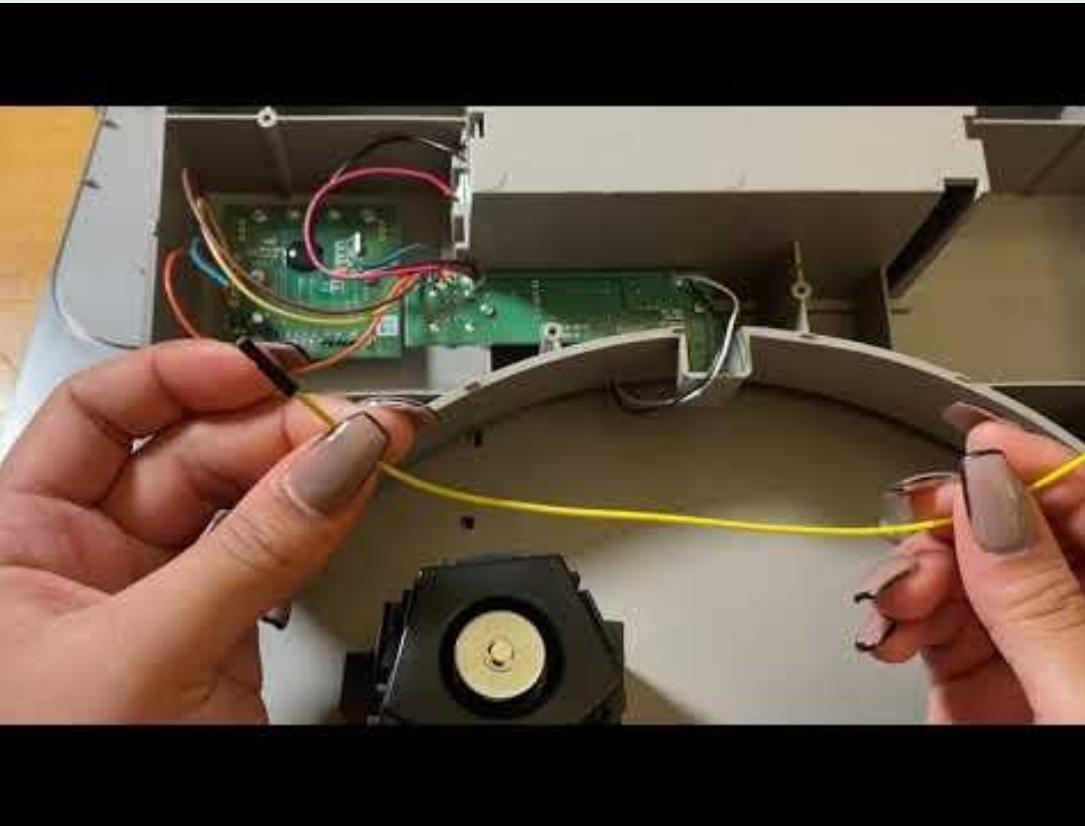
Connecting to an Arduino

```
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!)

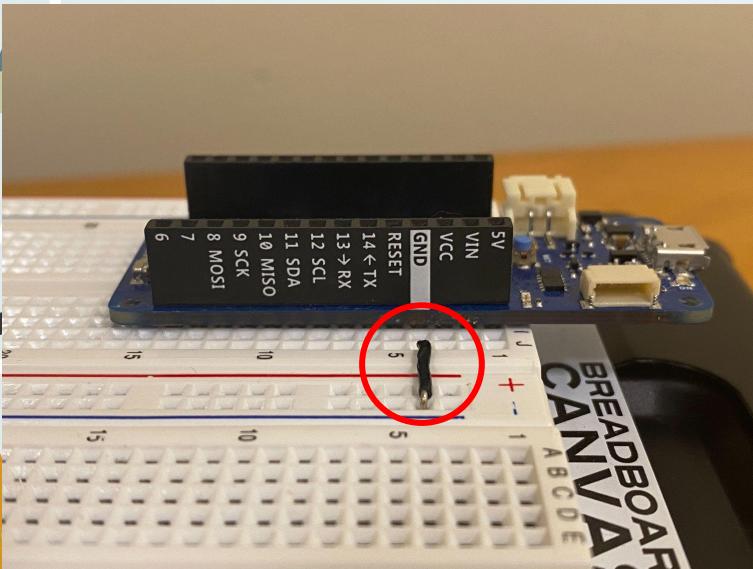
- We need to program for this Voltage drop
- In other words, when the Voltage goes from HIGH to LOW
- We can set the pins on the Arduino from HIGH to LOW on the Arduino Integrated Development Environment (IDE)
- We can set the voltage from an output digital pin (in this example I use pin 5)

Video Walkthrough

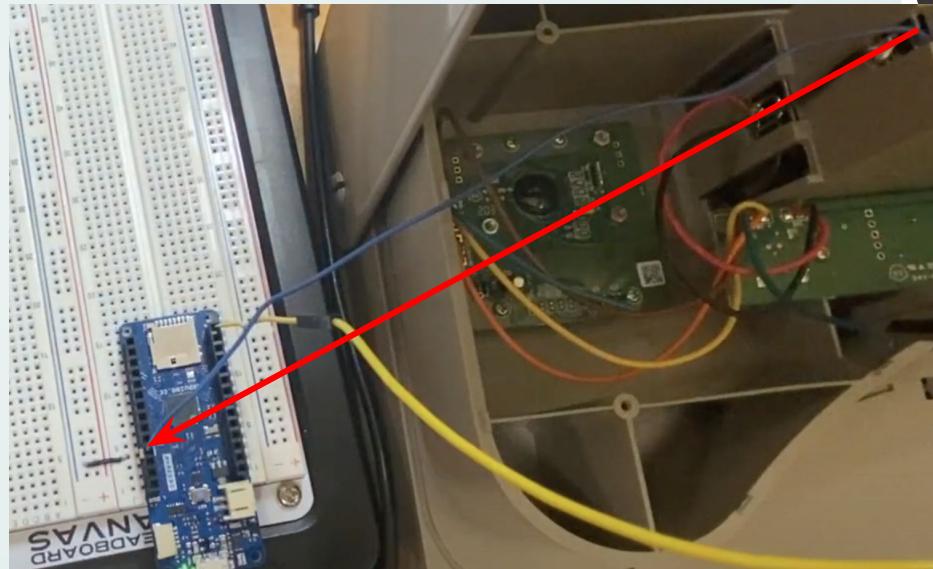


Debugging!

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



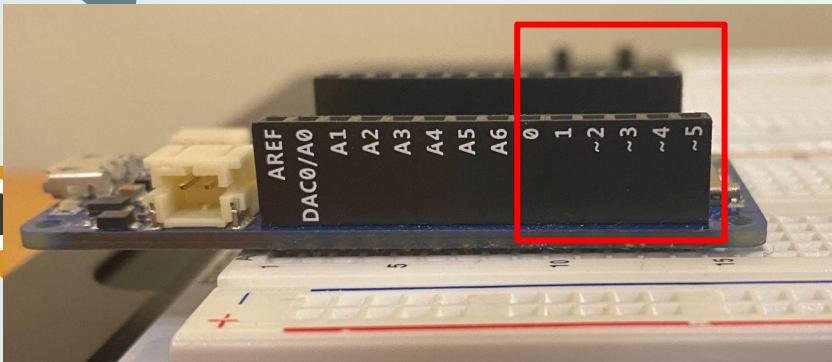
^Arduino Connection to GND



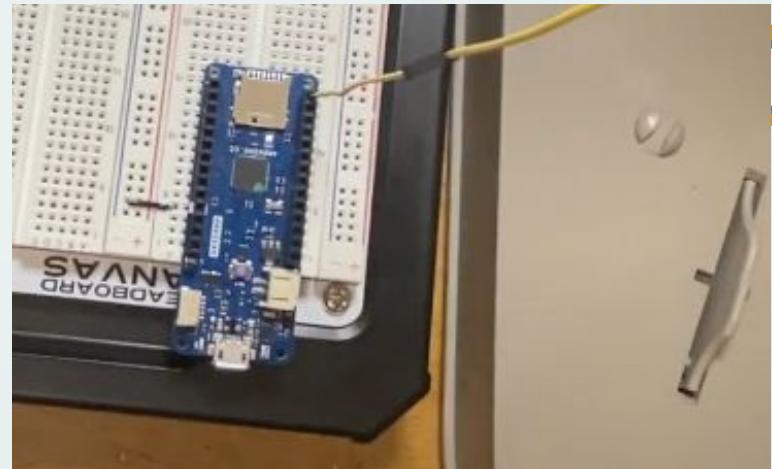
^Feeder Connection to GND (Blue Wire in Video)

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 §

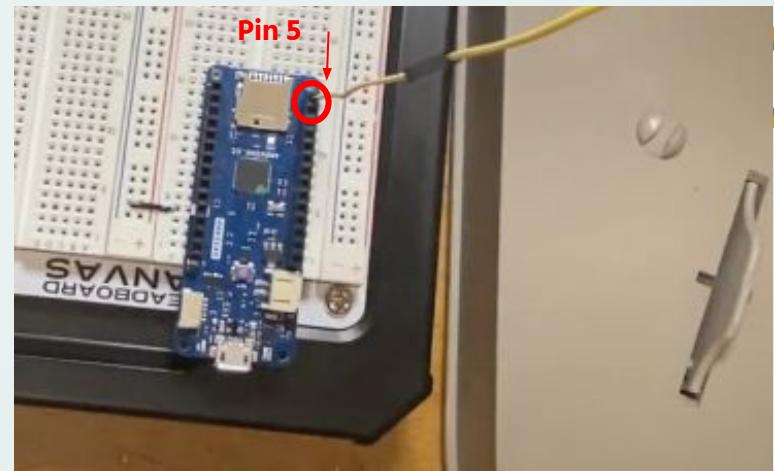
void setup() {
  // put your setup code here, to run once:
  pinMode(5, OUTPUT); // sets digital pin 5 as the output
}

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!



Thank You!