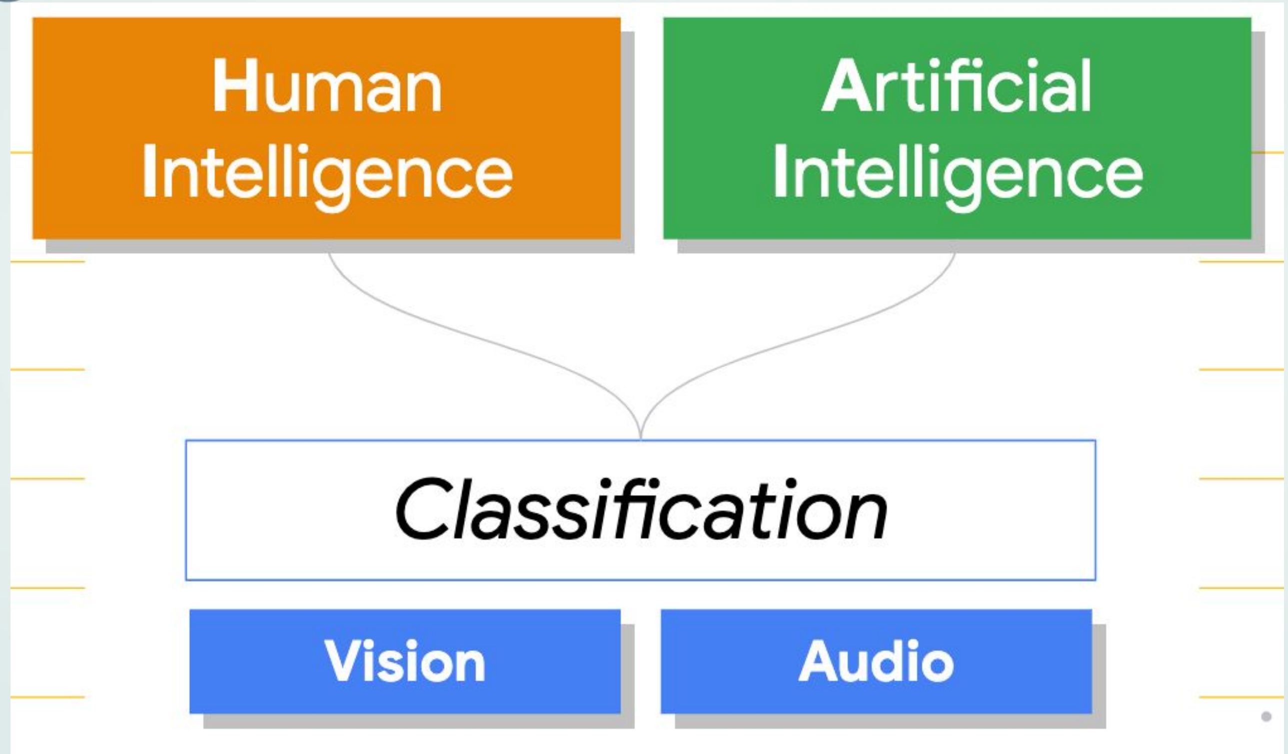
Using Keyword Spotting to Control Physical Systems

Ezenia Diaz-Lembert & Gage Hills
Harvard University

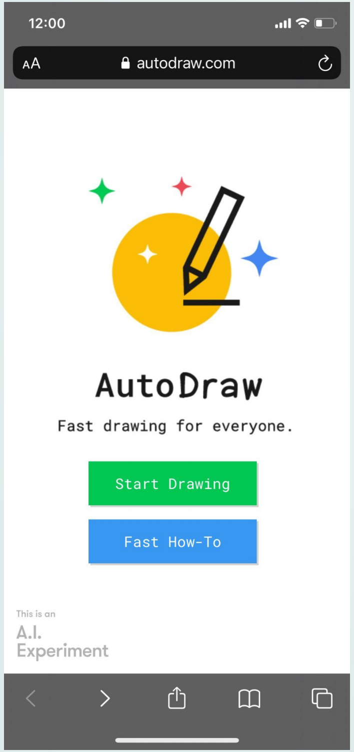
CRESTLEX3 Workshop, 2021
https://tinymlx.org/CRESTLEX3/

So Far

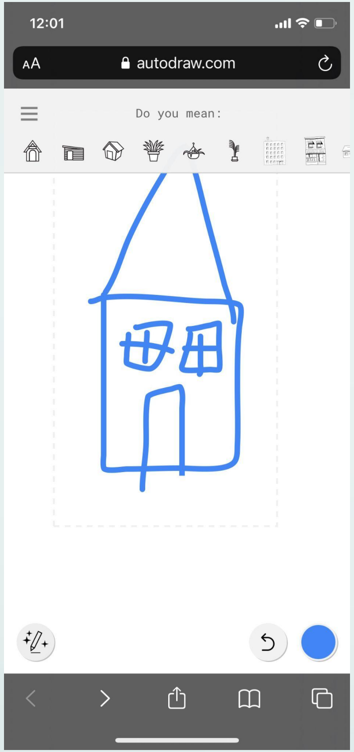


Human Intelligence
Artificial Intelligence
Classification
Vision
Audio

So Far



12:00
autodraw.com
AutoDraw
Fast drawing for everyone.
Start Drawing
Fast How-To
This is an
A.I.
Experiment

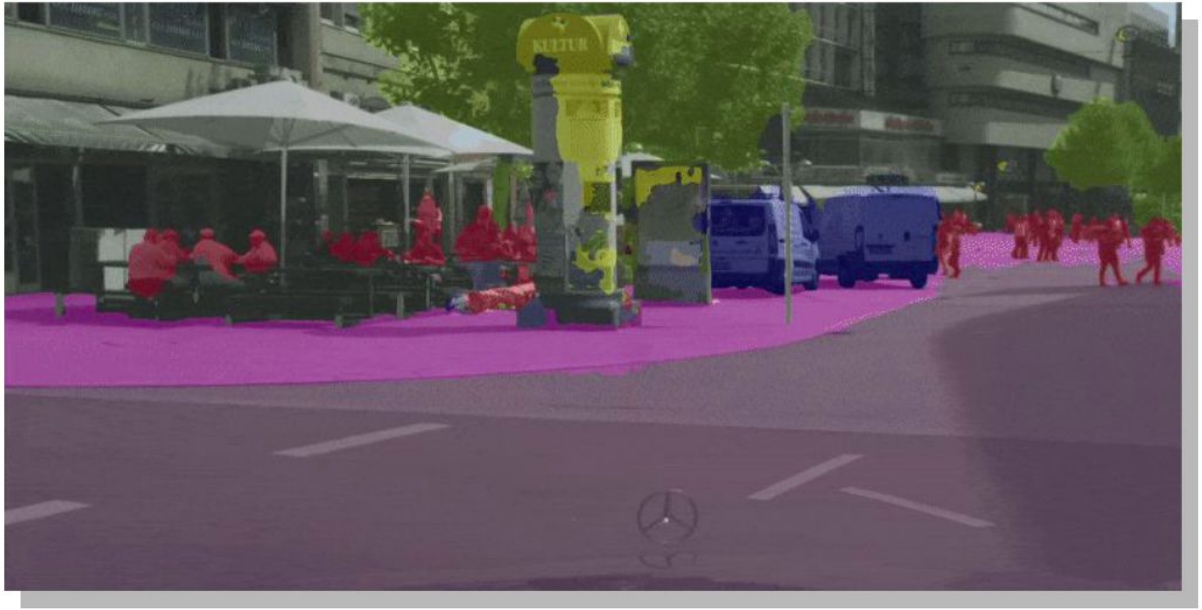


12:01
autodraw.com
Do you mean:

Gato
SPANISH
Cat
ENGLISH
Cat: 96%, Felidae: 90%, Carnivore: 89%

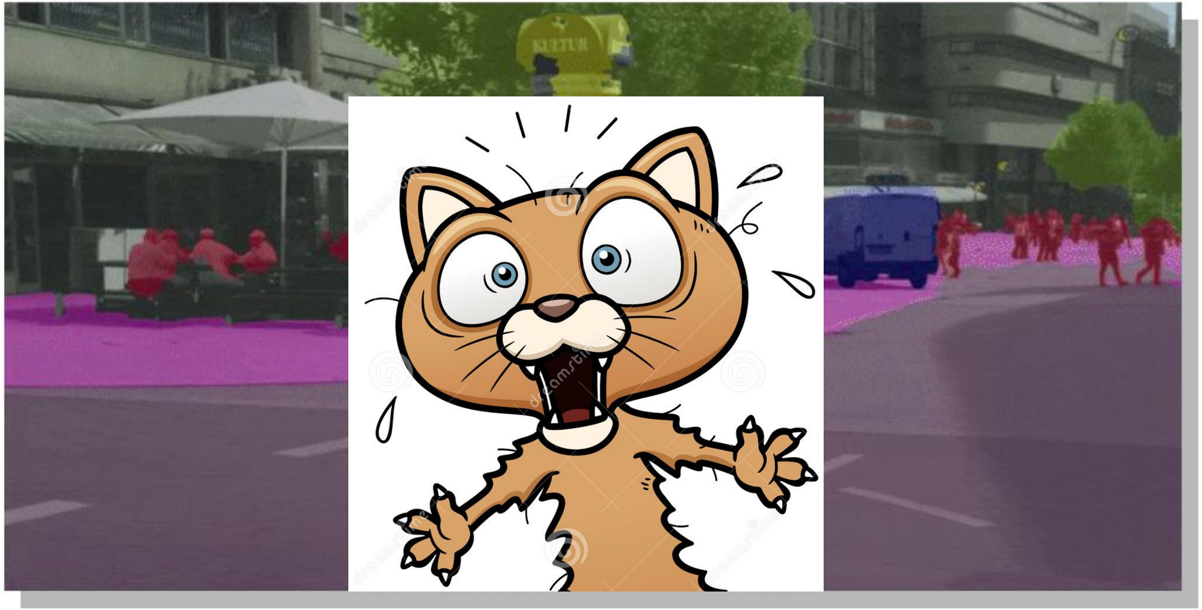
So Far

Segmentation



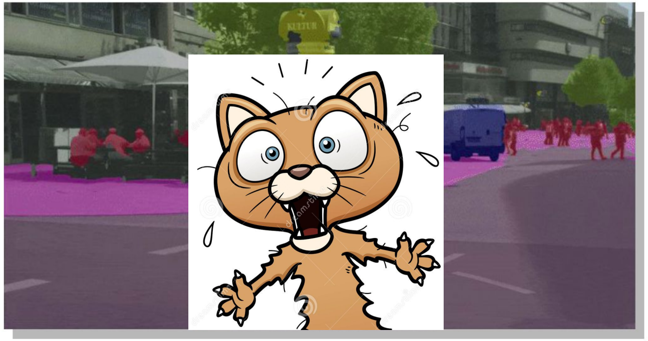
So Far

"Cat" detected



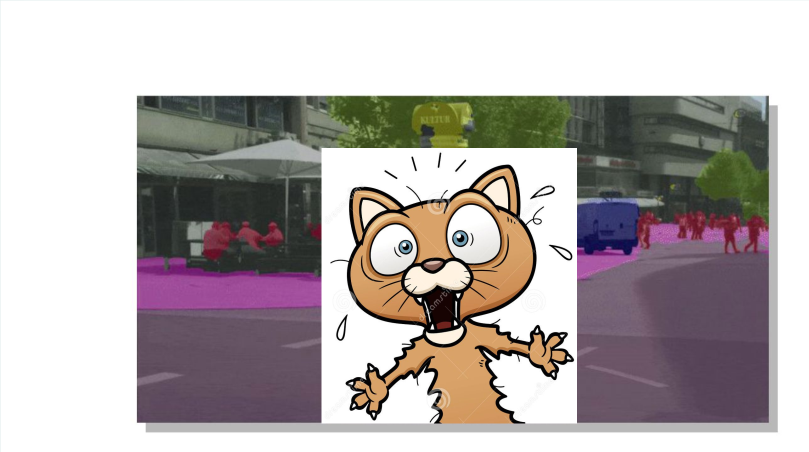
Then What ?

Cat detected...



Then What ?

Cat detected...



BRAKE!!!

Today, we're going to use

<u>Keyword Spotting</u>

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!

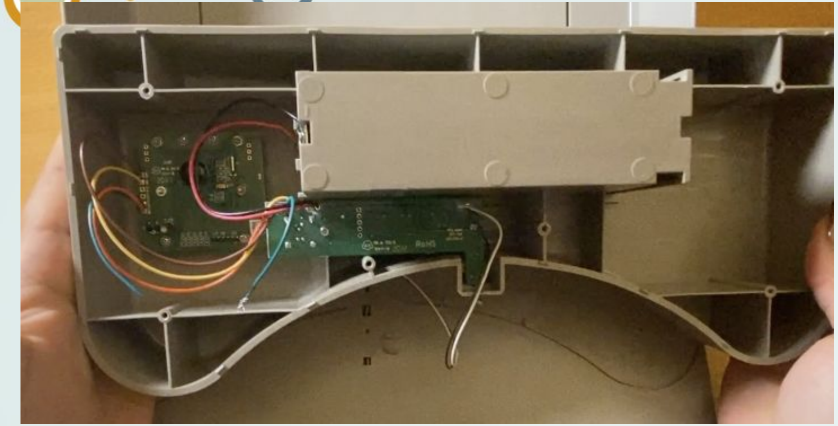
The Automatic Pet Feeder

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

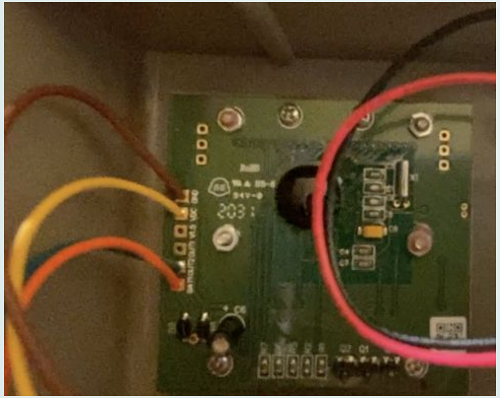
- But, how does the pet feeder know **when** to rotate ? How does it know how to rotate at all?

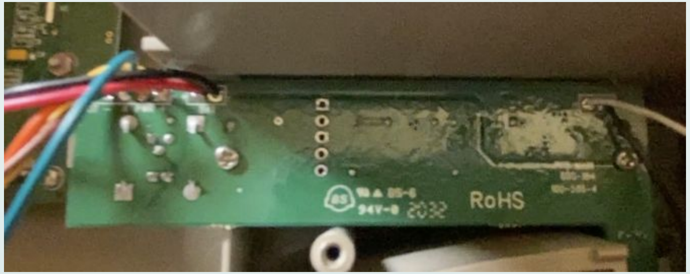
How Does it Work?



- 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



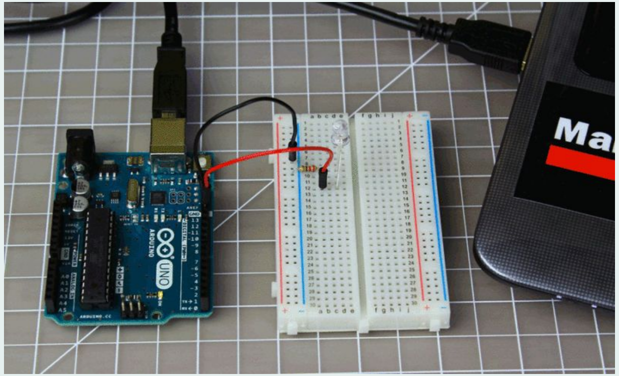


Today

Step 1:
Build TinyML model

EDGE IMPULSE

Step 2:
Run it on Hardware

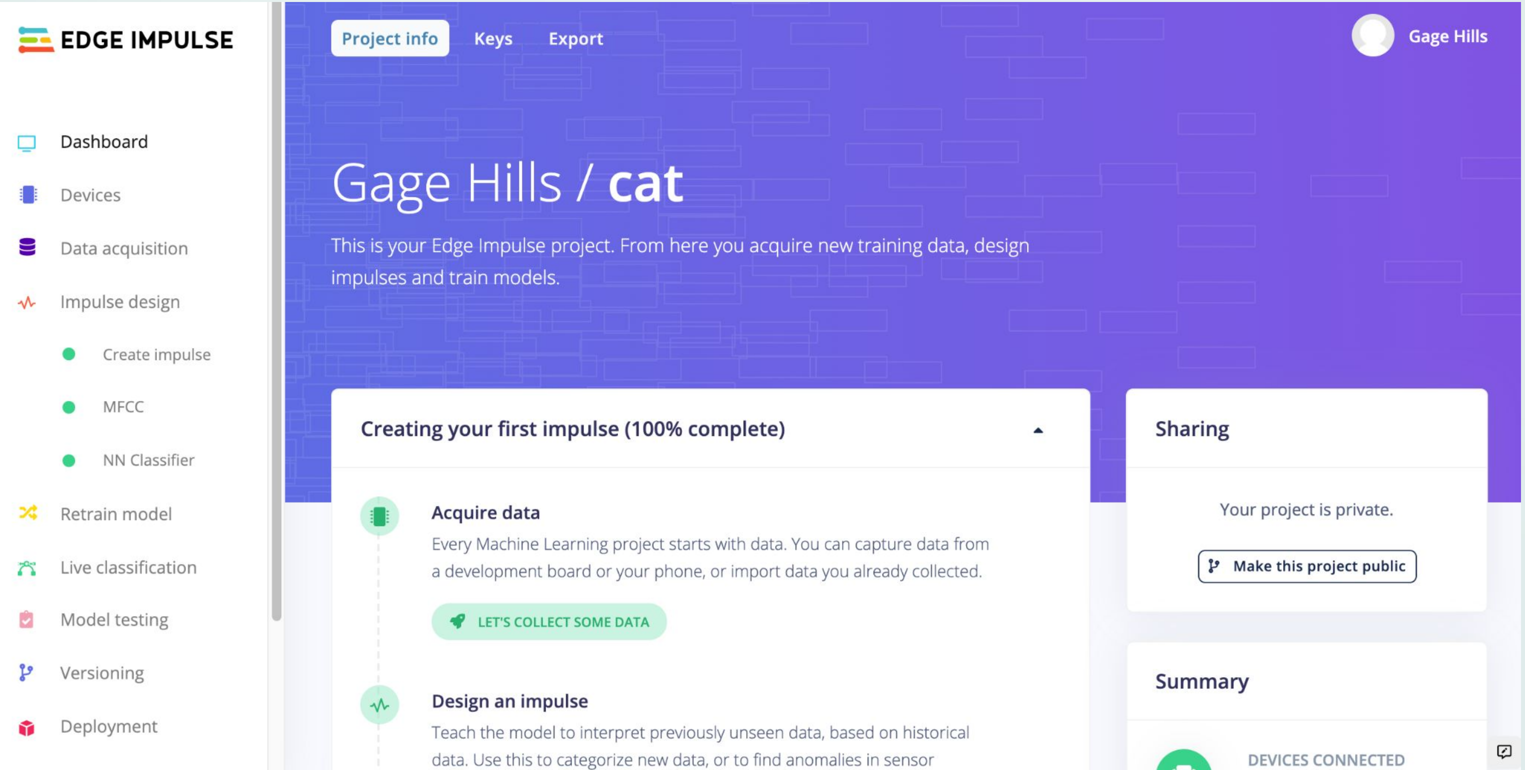


Arduino BLE Nano 33

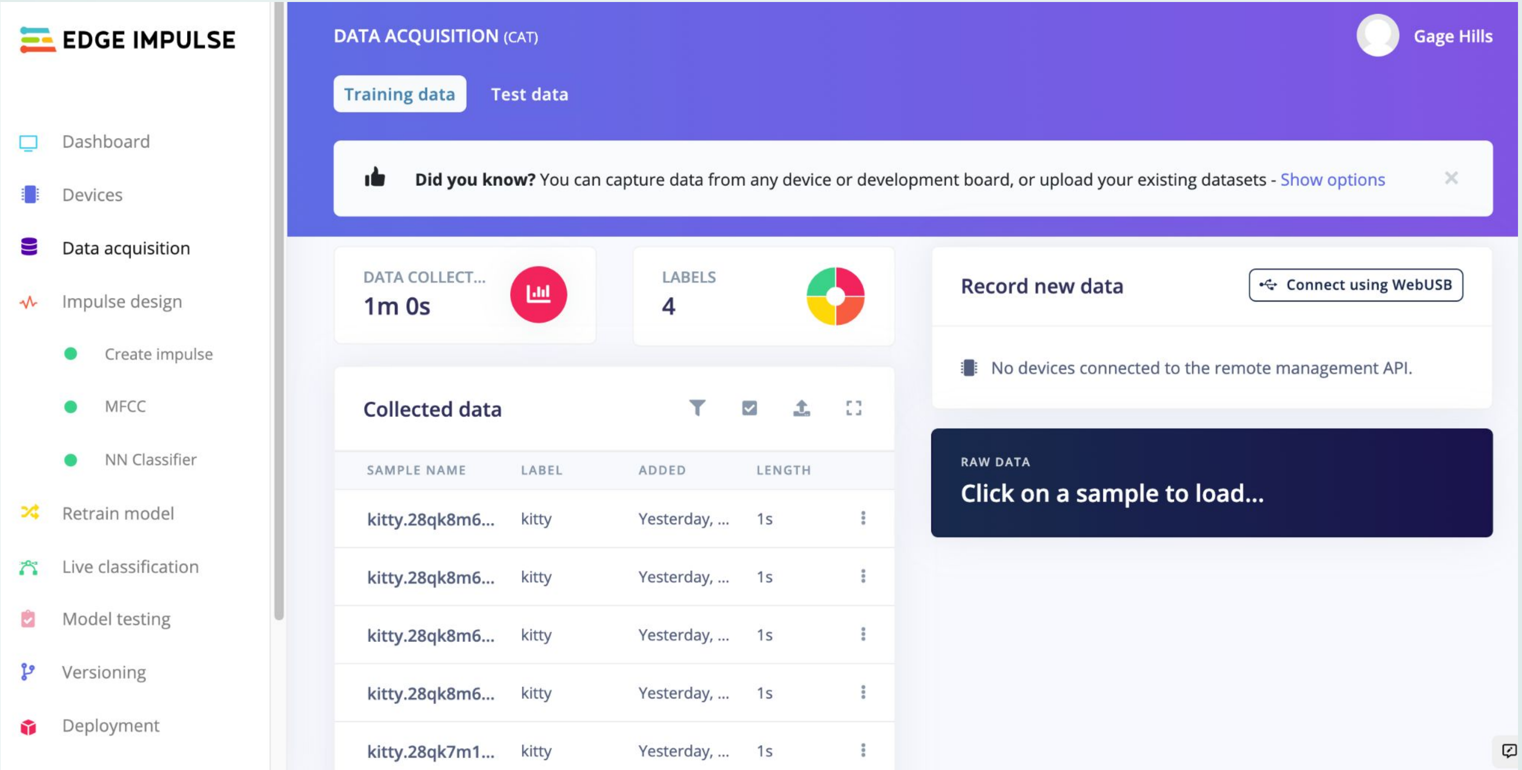
Step 3:
Feed my cats

Pet feeder

Step 1: Build TinyML Model

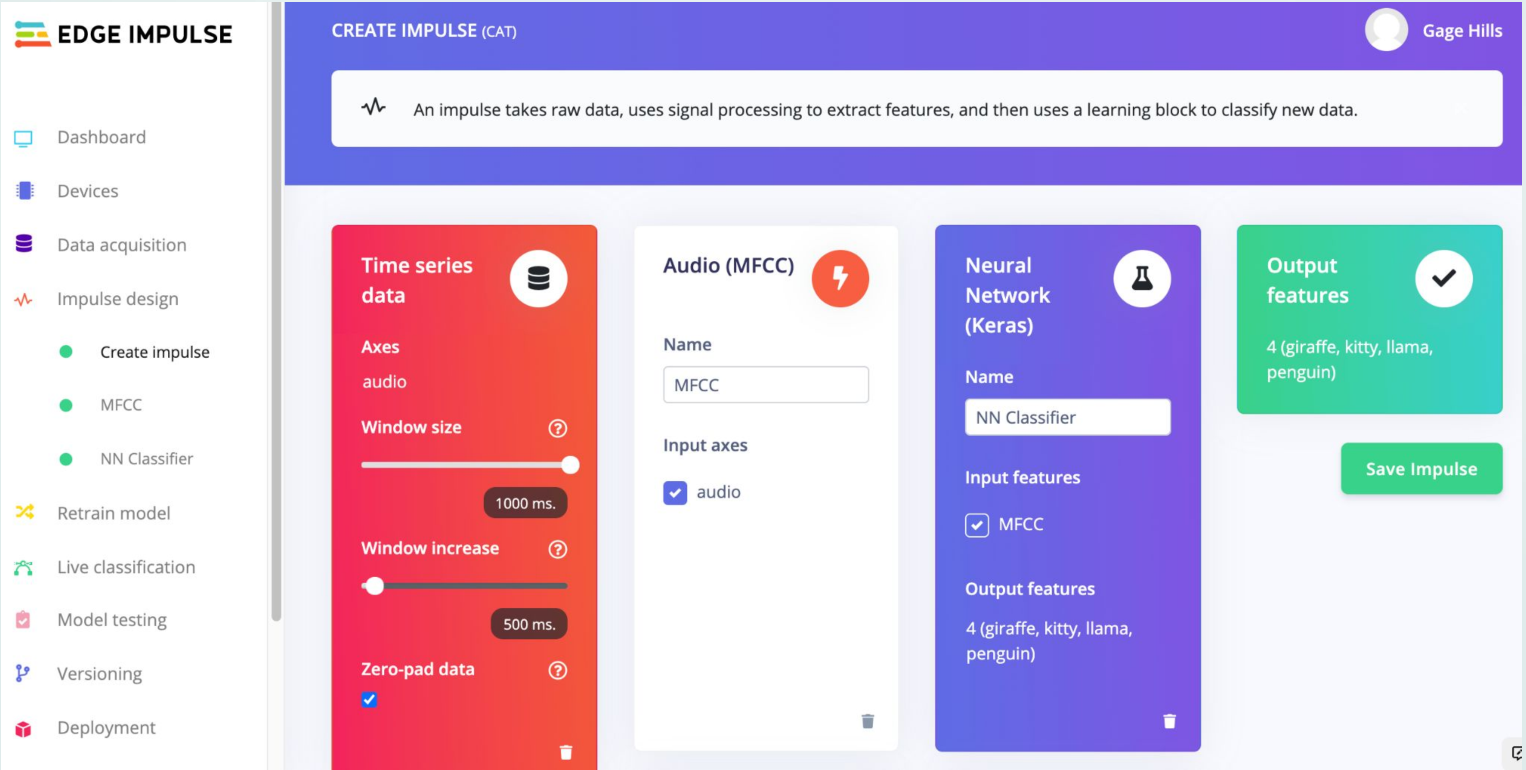


Step 1: Build TinyML Model

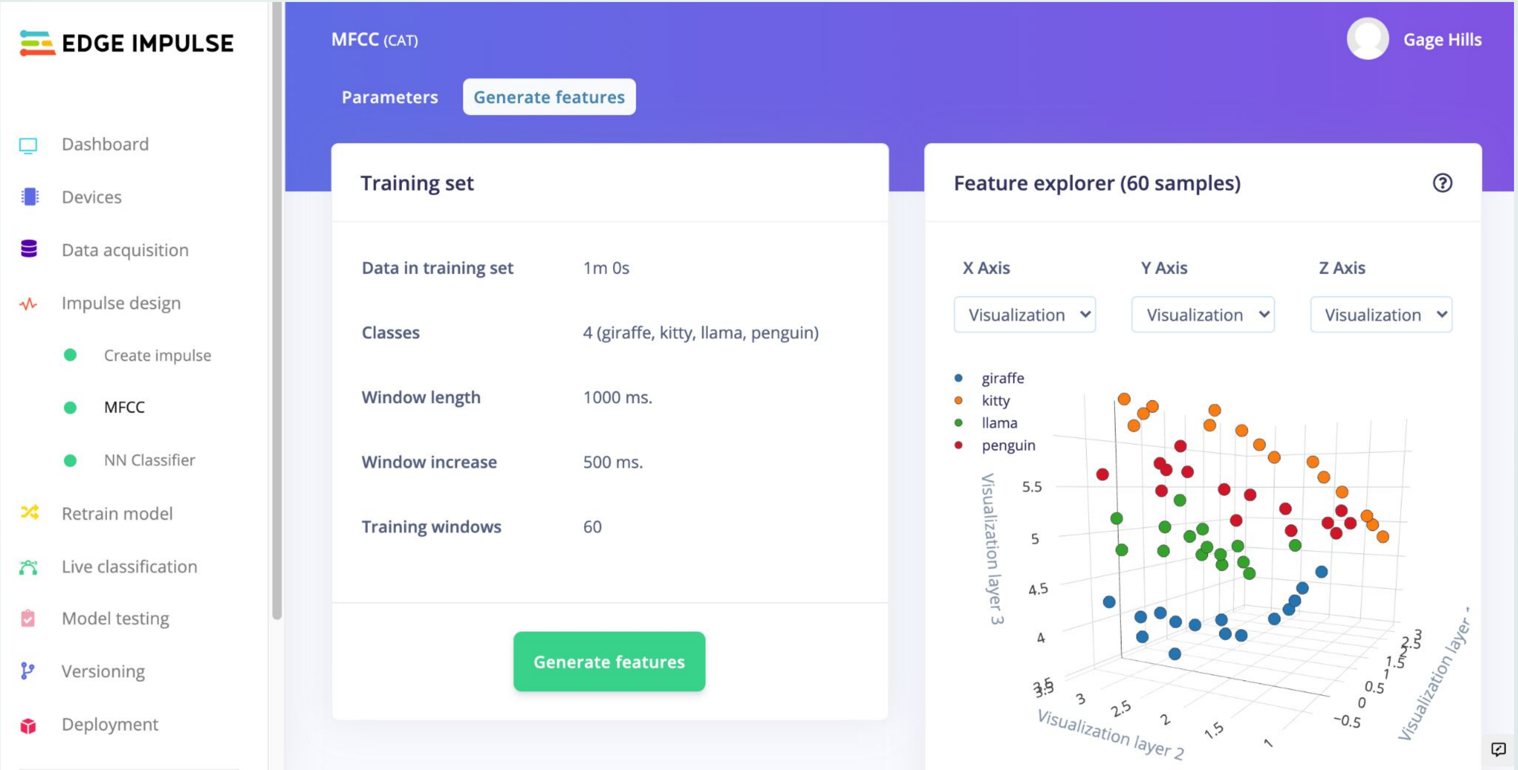


Step 1: Build TinyML Model

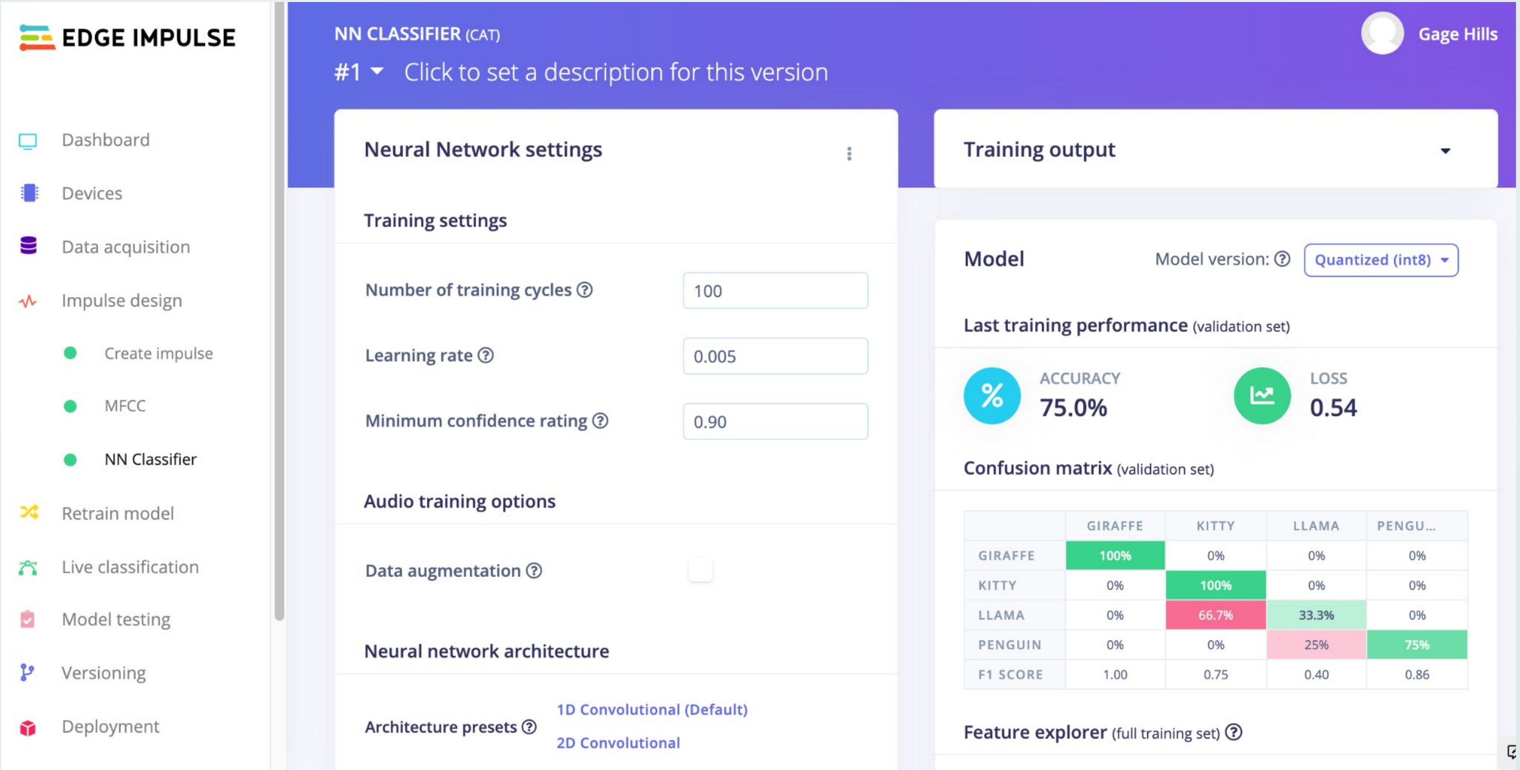
Step 1: Build TinyML Model



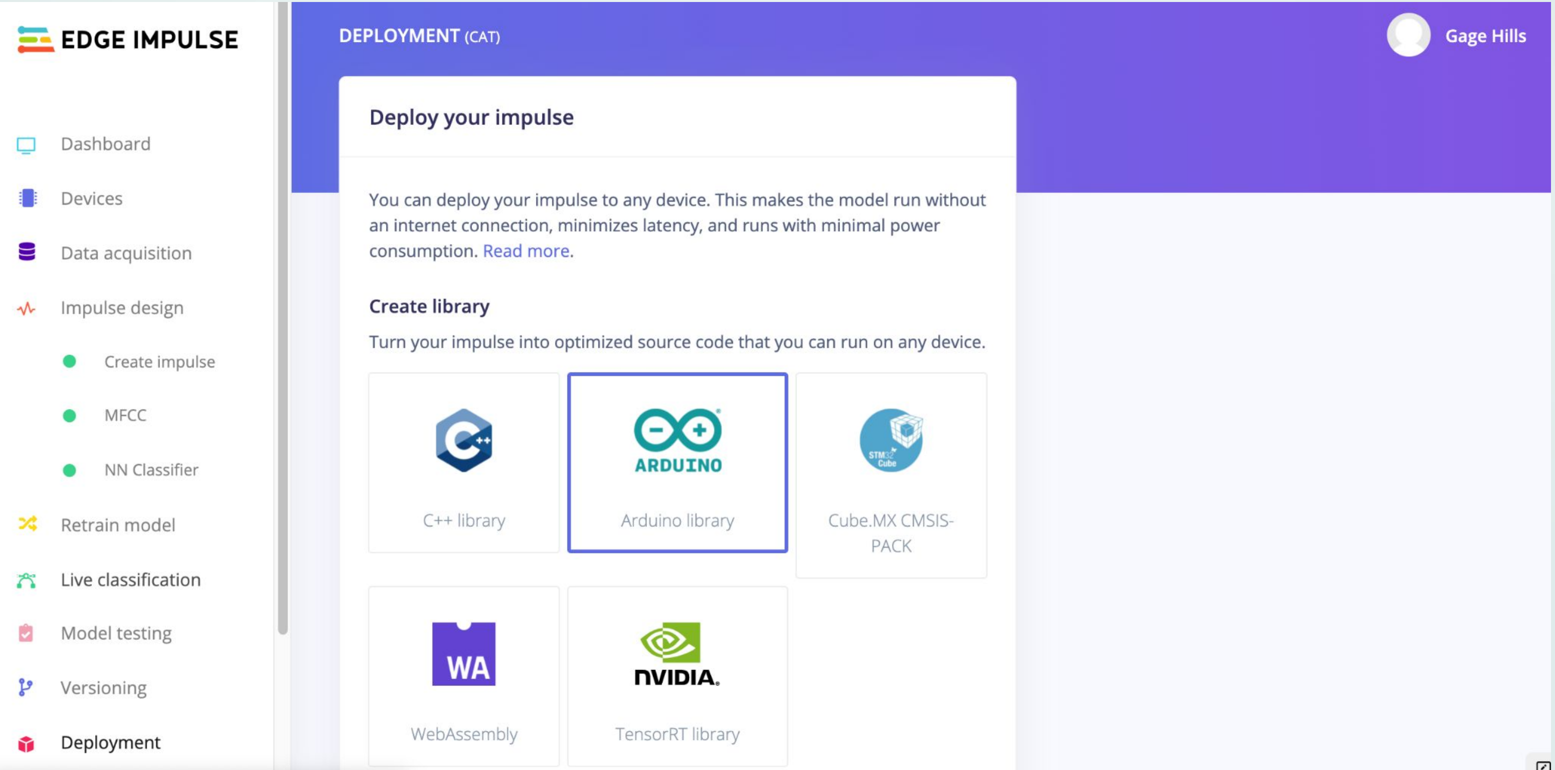
Step 1: Build TinyML Model



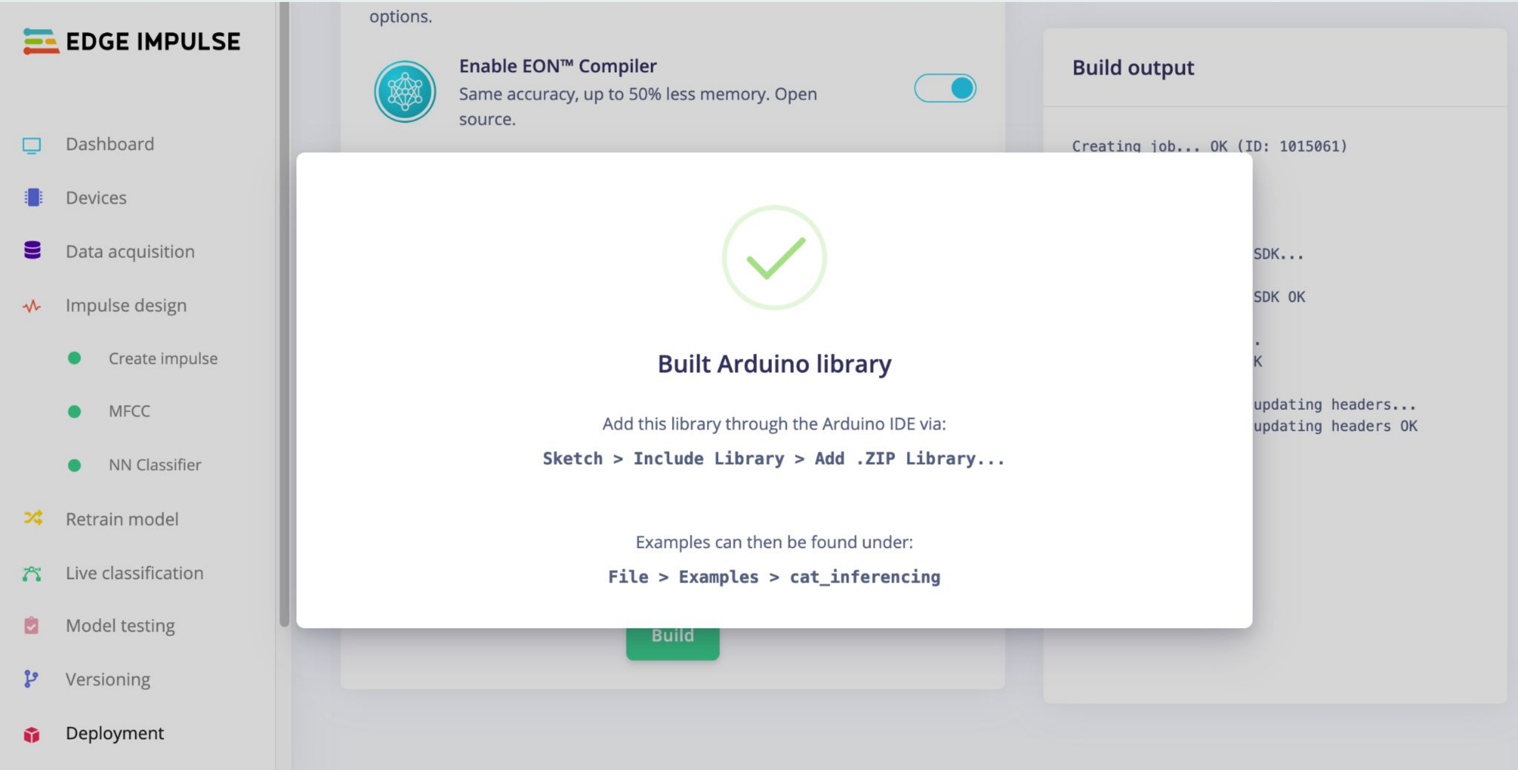
Step 1: Build TinyML Model



Step 2: Run it on Arduino



Step 2: Run it on Arduino



Step 2: Run it on Arduino

nano_ble33_sense_microphone | Arduino 1.8.15

nano_ble33_sense_microphone §

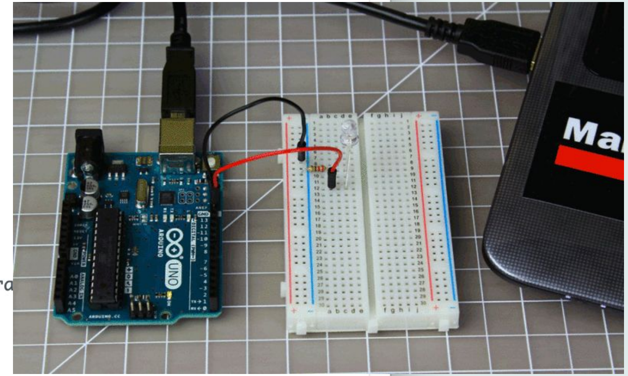
```
/* Includes ---------------------------------------------------------------------------- */
#include <PDM.h>
#include <cat_inferencing.h>

/** 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 ra

/**
 * @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

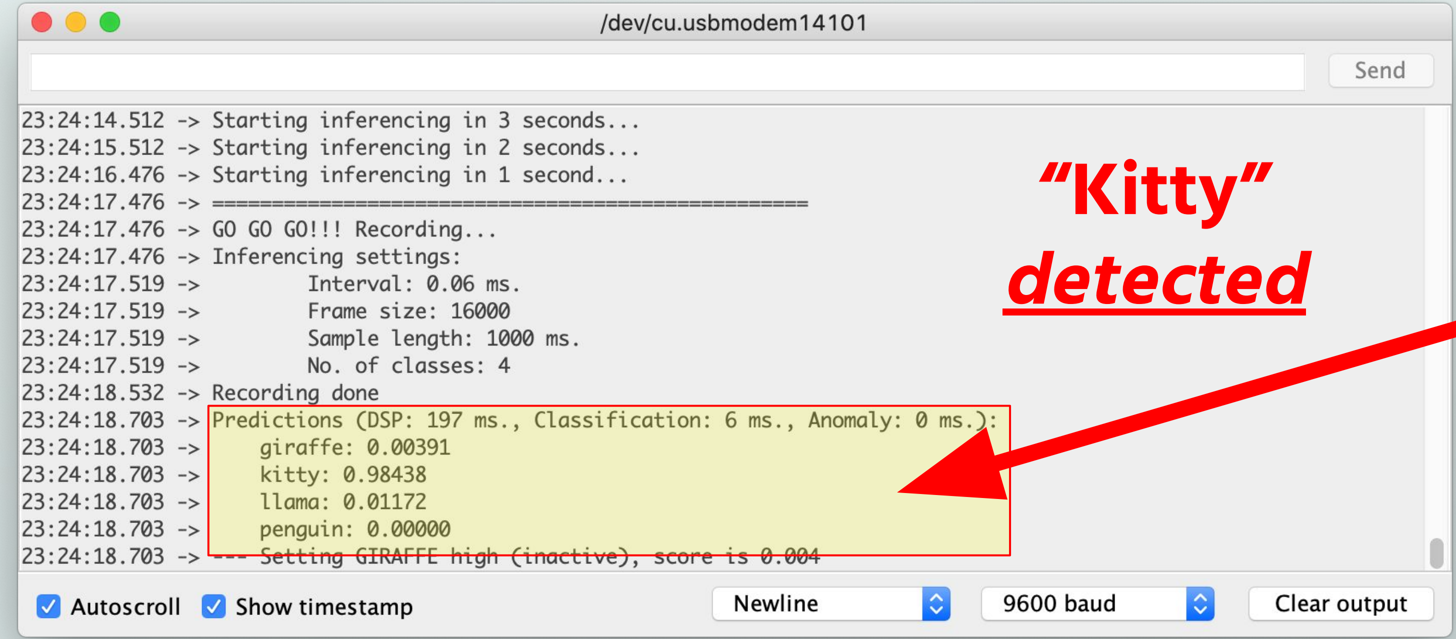
Send

```
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)

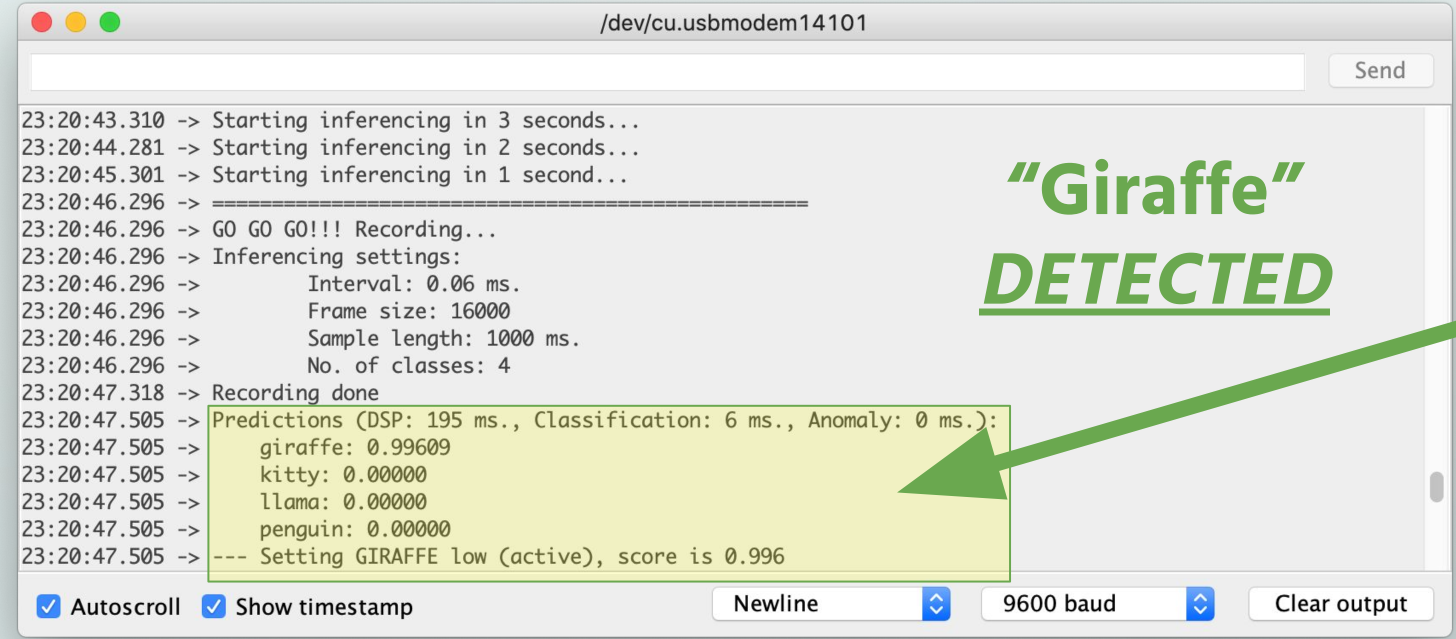


"Kitty"

detected

Step 2: Run it on Arduino

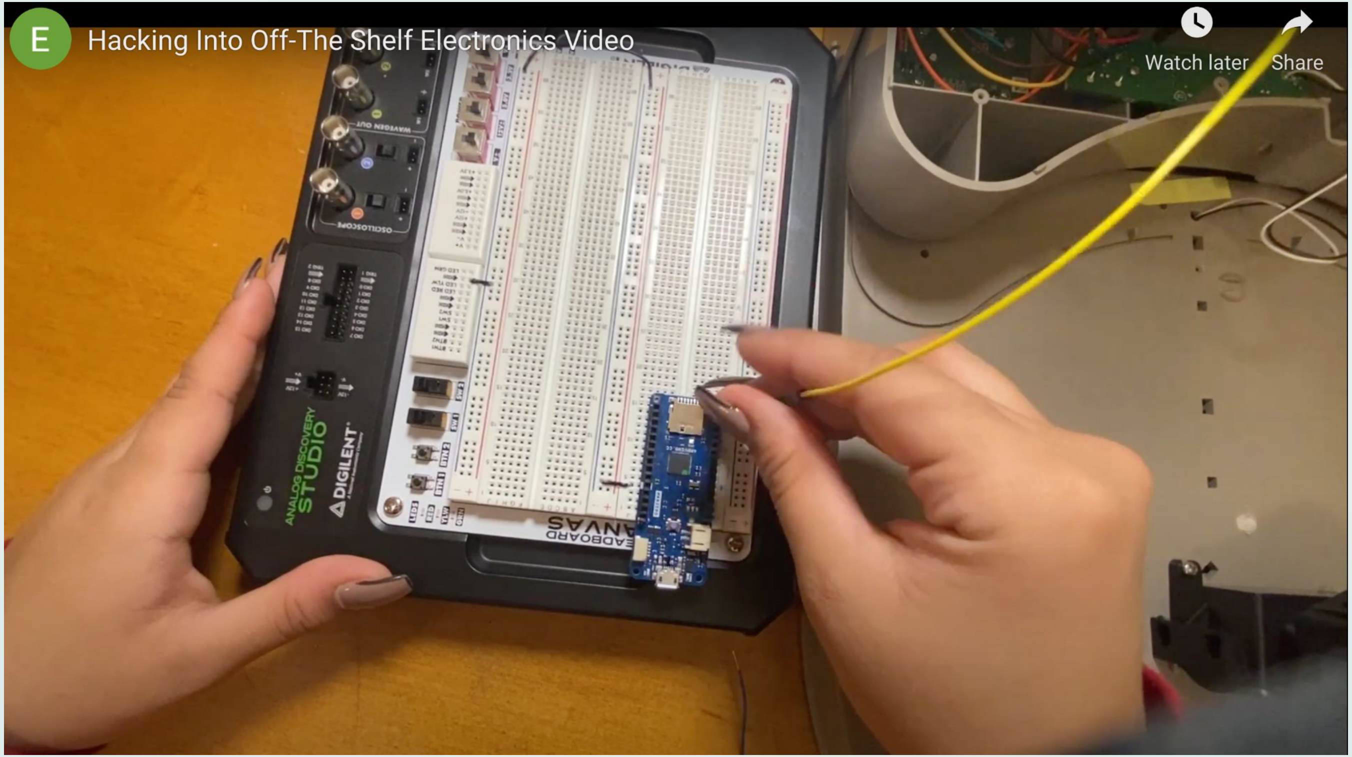
Status Output (for detailed program information)



"Giraffe"

DETECTED

Step 3: Connect it to Pet Feeder



LIVE DEMO

featuring:

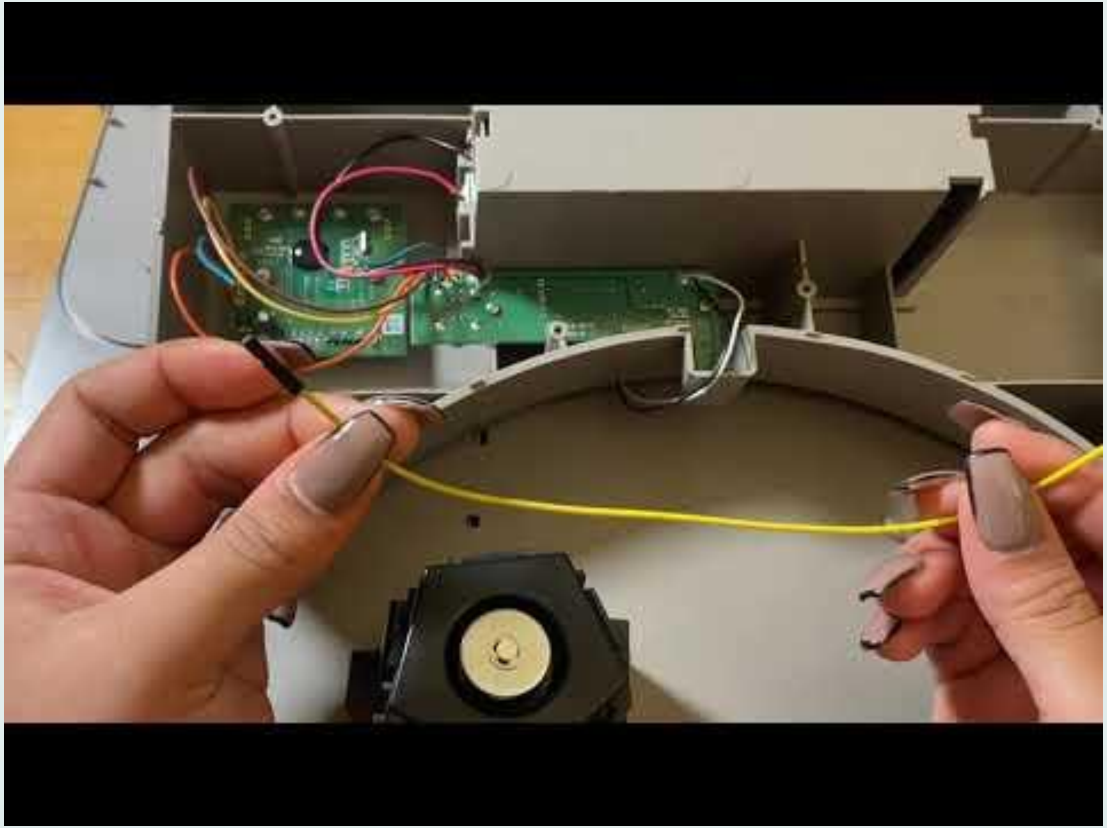
Chuckles & Cakmak-boy

VIDEO WALKTHROUGH

starring:

Ezenia Diaz-Lembert

Video Walkthrough: Pet Feeder + Arduino

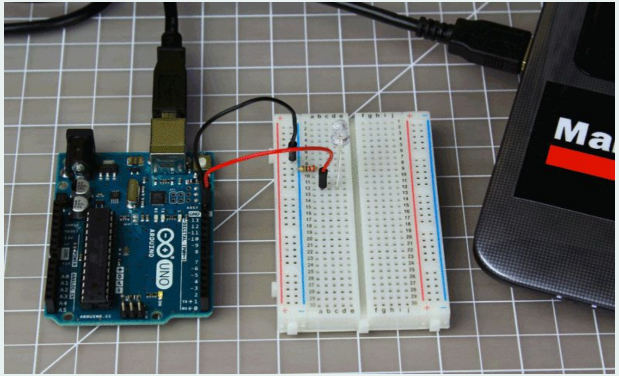


Questions ?

Step 1:
Build TinyML model

EDGE IMPULSE

Step 2:
Run it on Hardware



Arduino BLE Nano 33

Step 3:
Feed my cats

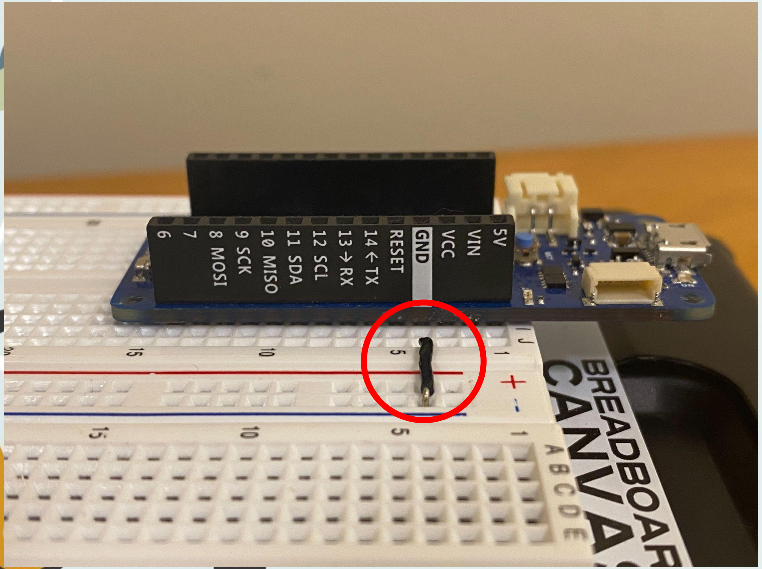
Pet feeder

Thank You!

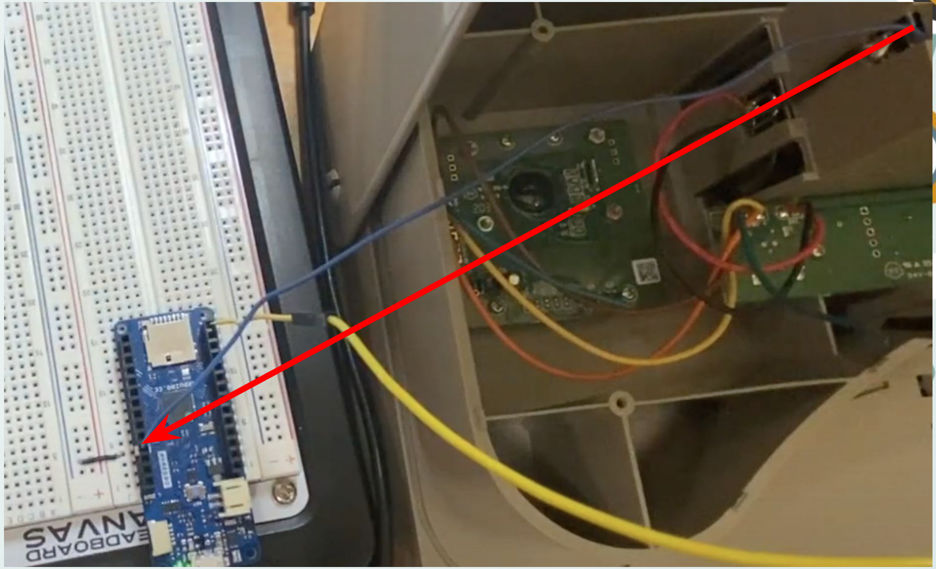
Back-up slides

Debugging!

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



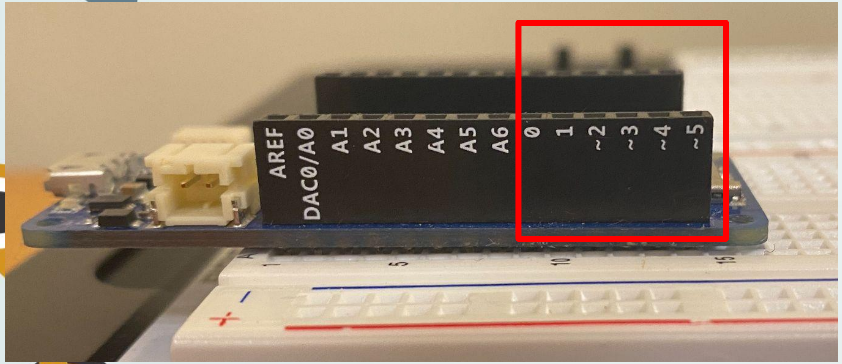
^Arduino Connection to GND



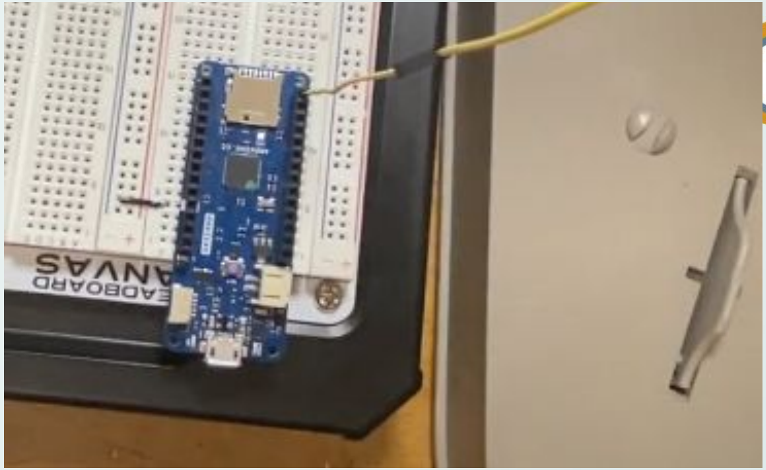
^Feeder Connection to GND (Blue Wire in Video)

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!



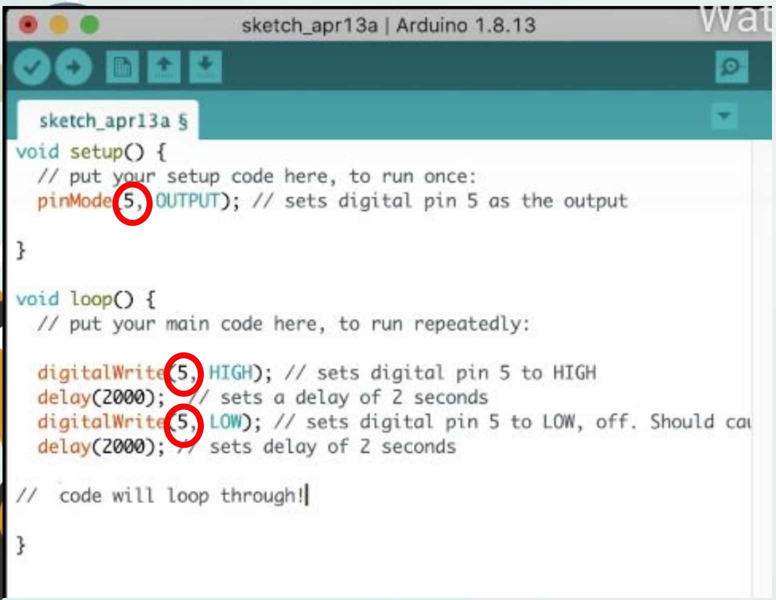
^Digital Pins



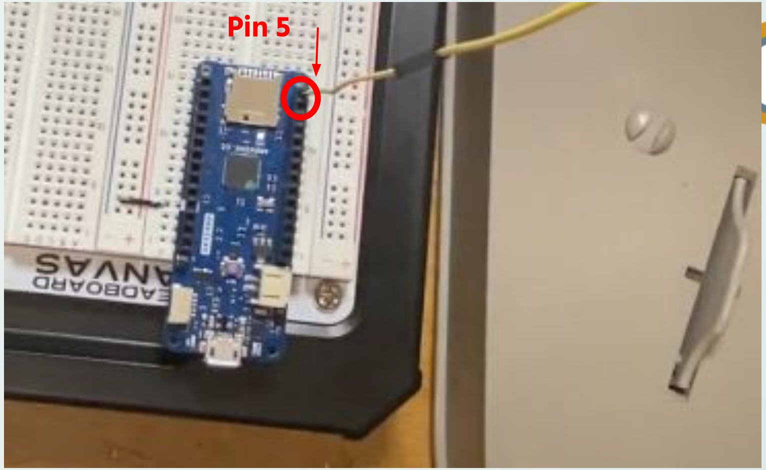
^Feeder Connection to PWR (Yellow Wire in Video)

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!



^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!

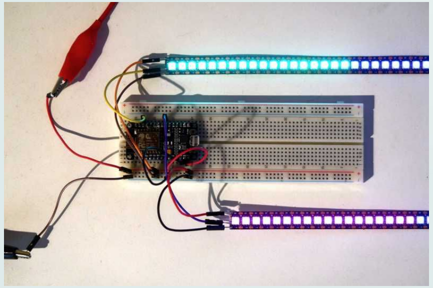
Hacking off Into Off-The Shelf Electronics

Project: Automatic Pet Feeder

Applications

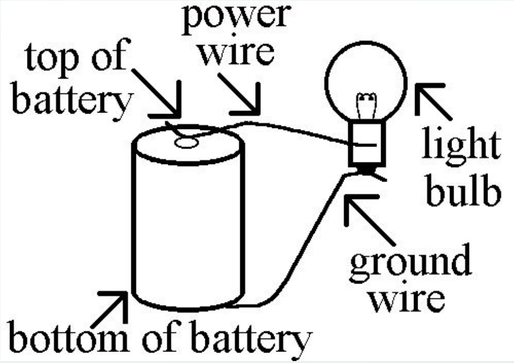
- Much of the same ML and Arduino concepts you’ve been learning about are seen in electronics all around us!

- 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.

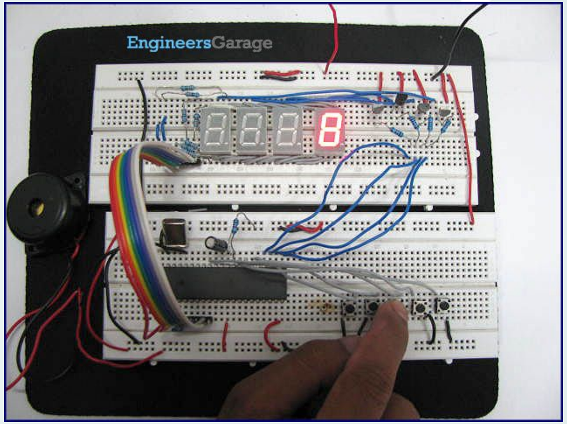


Applications

- 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!

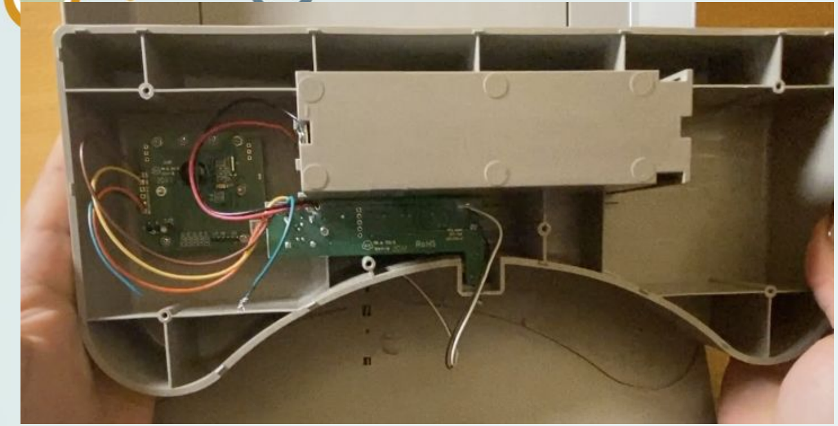
The Automatic Pet Feeder

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

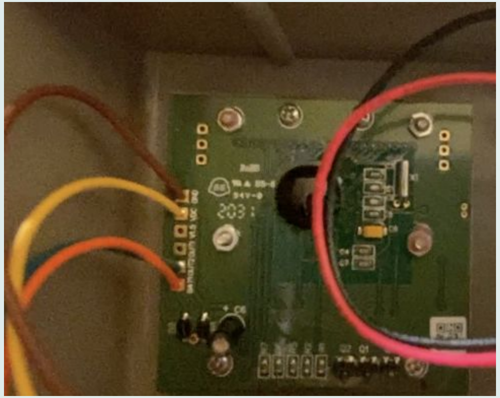
- But, how does the pet feeder know **when** to rotate ? How does it know how to rotate at all?

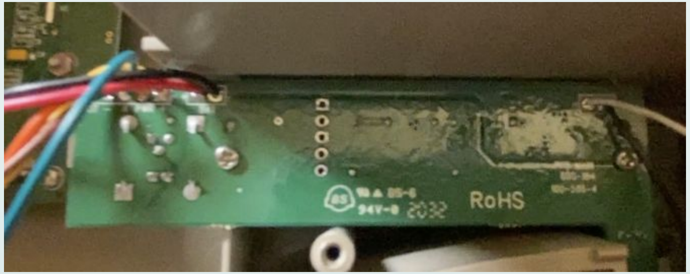
How Does it Work?



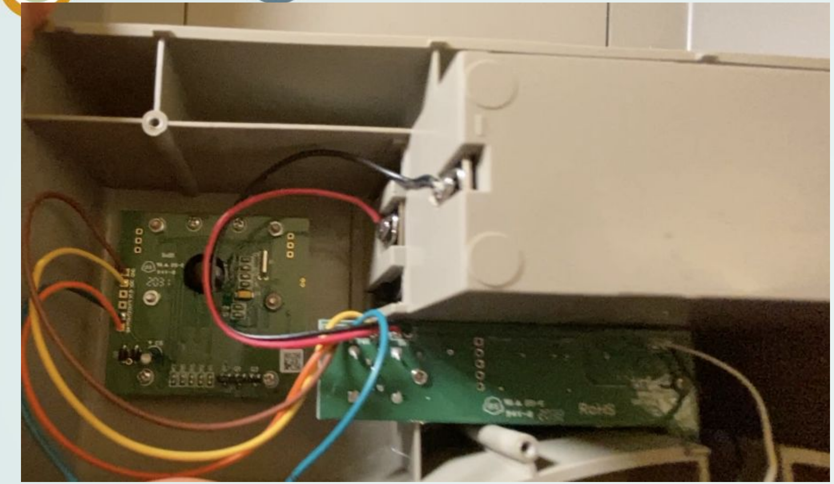
- After unscrewing the back of the pet feeder, you should see something like this beneath the control system of the feeder

- 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?



$$P = IV$$

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!

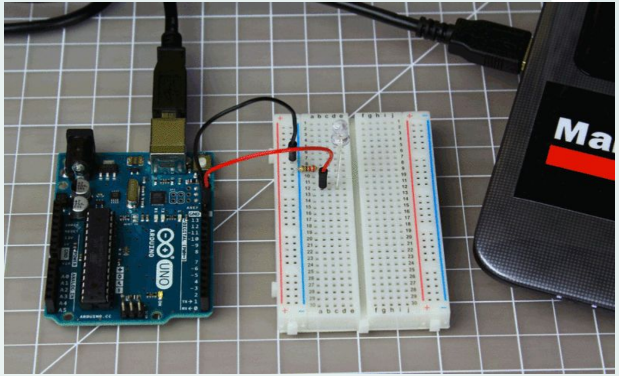
Wires:

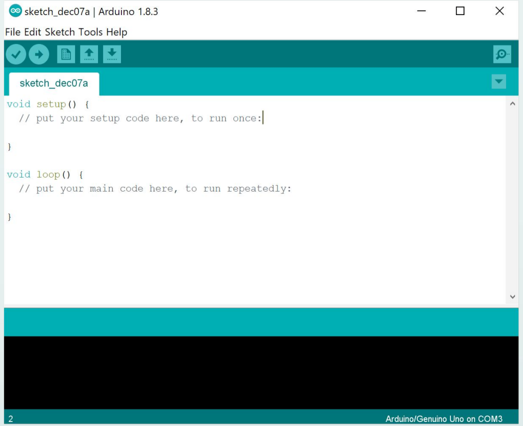
- Black and White → Motor
- Yellow, Brown, Green → Digital Control Board
- VVC, GND → Power for digital chip
- Motor needs a higher voltage, control signal given by OUTZ.
- For us to control the signal for motor movement, we should cut OUTZ (Blue).

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.





Connecting to an Arduino

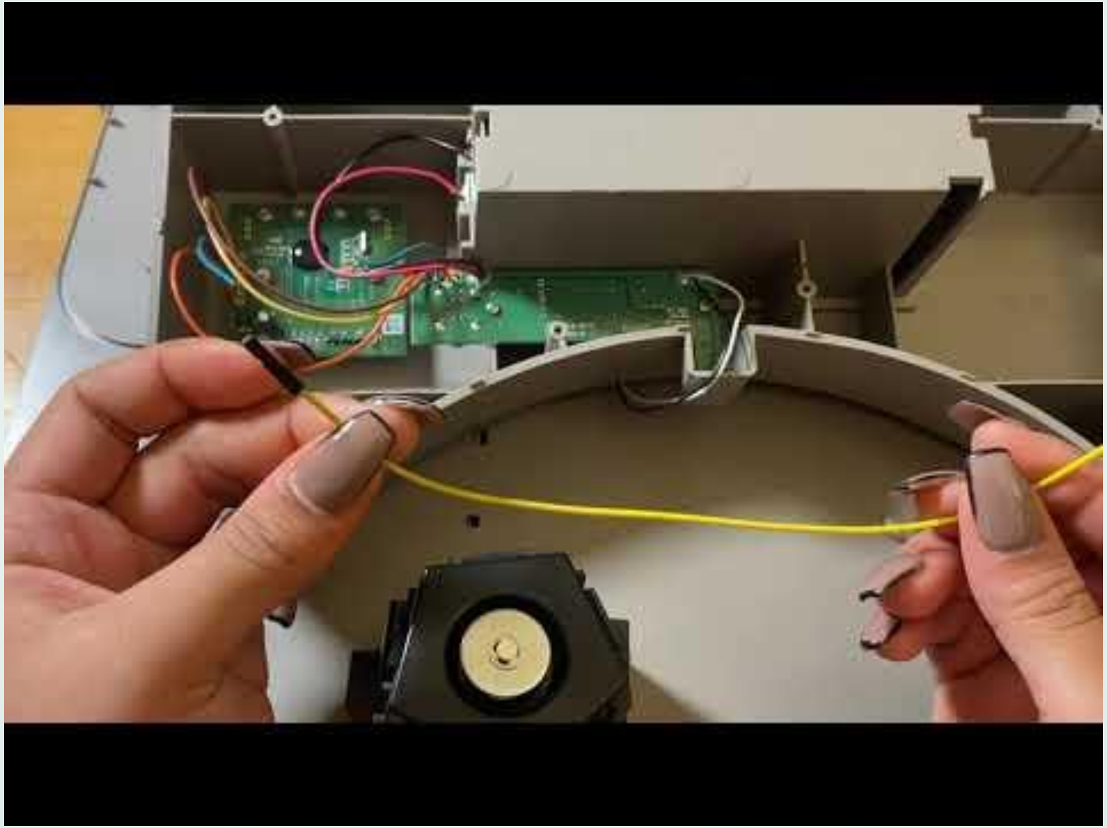
```
void setup() {
  pinMode(5, OUTPUT);    //  digital pin 5 is the output
}

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!)

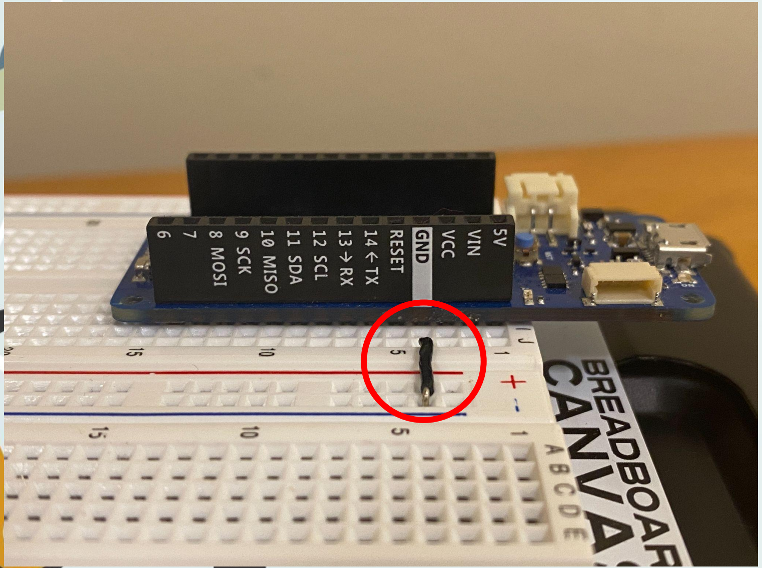
- We need to program for this Voltage drop
- In other words, when the Voltage goes from HIGH to LOW
- We can set the pins on the Arduino from HIGH to LOW on the Arduino Integrated Development Environment (IDE)
- We can set the voltage from an output digital pin (in this example I use pin 5)

Video Walkthrough

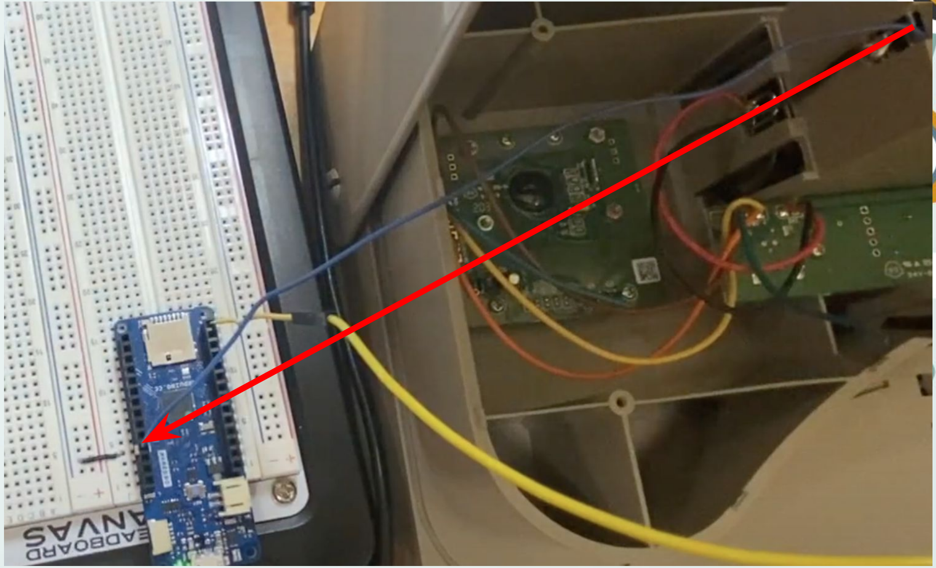


Debugging!

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



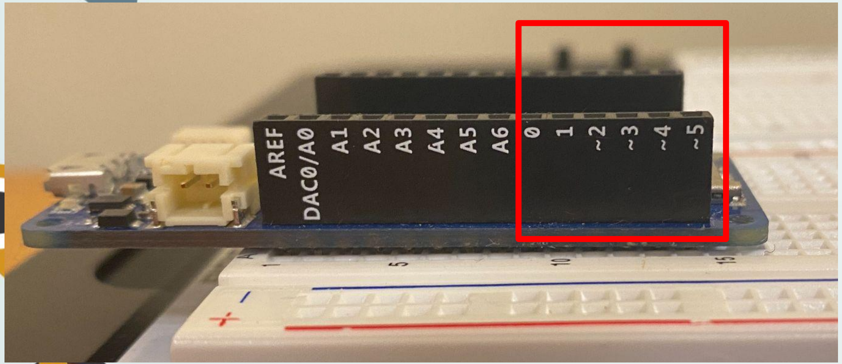
^Arduino Connection to GND



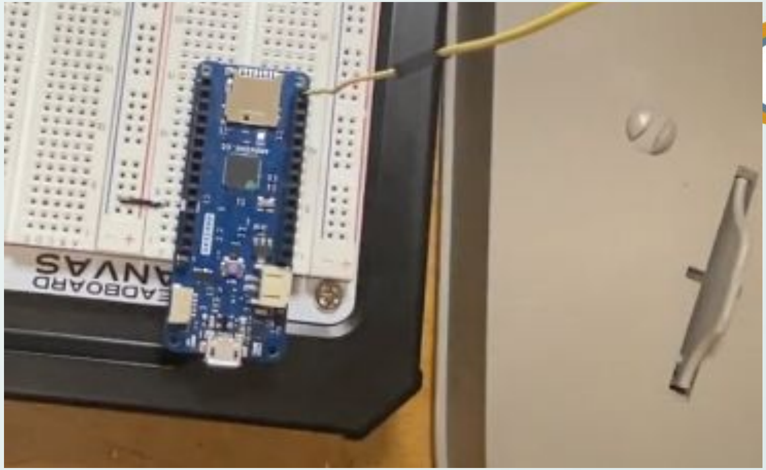
^Feeder Connection to GND (Blue Wire in Video)

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!



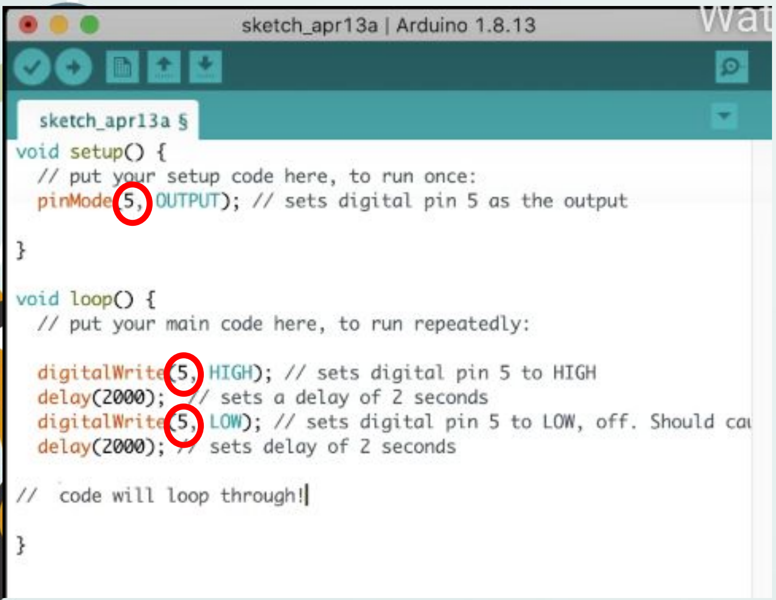
^Digital Pins



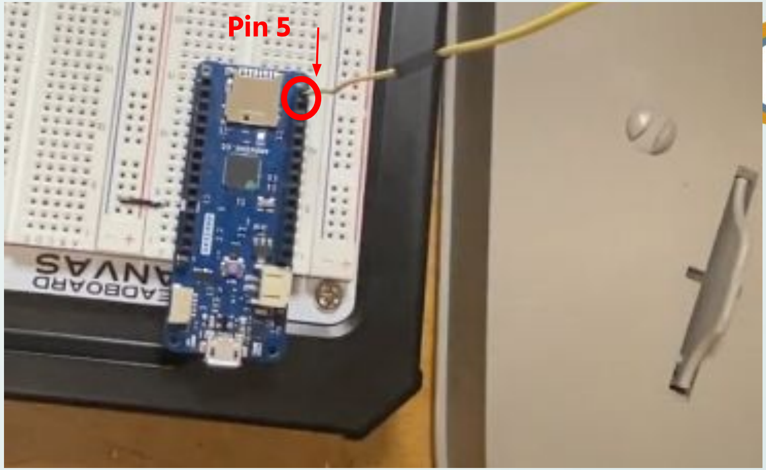
^Feeder Connection to PWR (Yellow Wire in Video)

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!



^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!