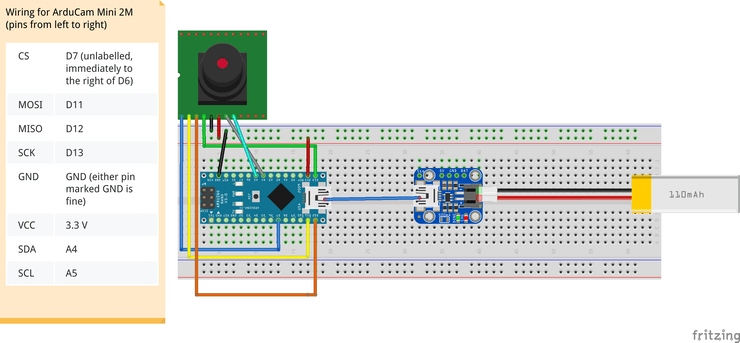
**Basic wiring**

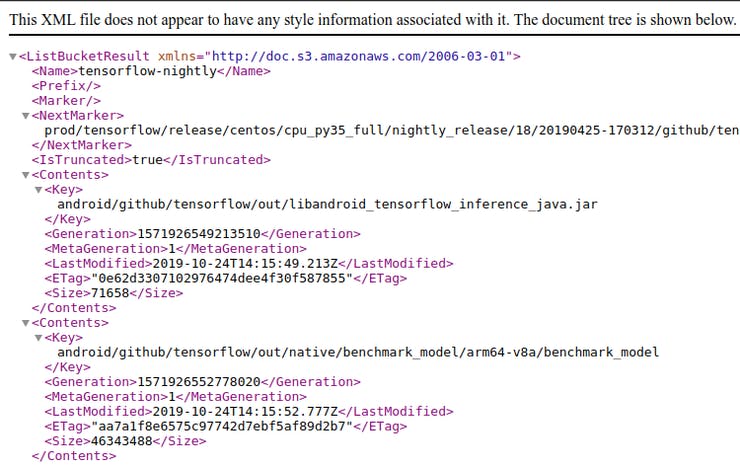
Follow the following wiring diagram to connect your Arduino Nano 33 BLE Sense to the ArduCam Mini 2MP. This diagram also shows how to add a lipo battery after you've flashed the Arduino.



pins on camera are in order from left to right

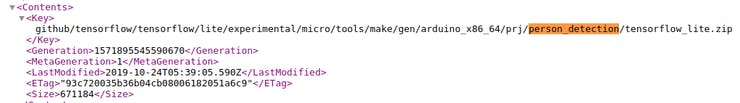
**Flashing the model**

A few examples of tiny models have been released on the [official TensorFlow repo](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro/examples/person_detection), the micro\_speech model being the most widely shown example. We wanted to flash the vision (person\_detection) demo, however as of the time of this post being written, not all of the resources are available. From the micro\_speech example, we saw that these Arduino zip packages were being stored in the same place as Google's [nightly TensorFlow builds](https://storage.googleapis.com/tensorflow-nightly/). When you navigate to this link, you'll see a document tree with links to all kinds of nightly builds:



document tree with links

Doing a quick word search through the document tree, we were able to locate the missing package! *(Update: TF team is working on updating the missing link)*



person\_detection found! Zip package under a slightly different name

Download this [package link](https://storage.googleapis.com/tensorflow-nightly/github/tensorflow/tensorflow/lite/experimental/micro/tools/make/gen/arduino_x86_64/prj/person_detection/tensorflow_lite.zip) and rename it to person\_detection.zip. This will be the package we upload as a library to our Arduino IDE as the [official instructions](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro/examples/person_detection#running-on-arduino) describe.

After uploading this zip package, we want to make a couple more modifications for things to run smoothly. With your text editor of choice, find where the Arduino library installed and open the library.properties file for editing

# In our case, our Arduino library is in ~/Arduino/libraries  
$ cd ~/Arduino/libraries/person\_detection  
$ vim library.properties

Change the first line that declares the name of the library to TensorFlowLite:person\_detection. Your file should look like this:

name=TensorFlowLite:person\_detection  
version=1.14-ALPHA  
author=TensorFlow Authors  
maintainer=Pete Warden <petewarden@google.com>  
sentence=Allows you to run machine learning models locally on your device.  
paragraph=This library runs TensorFlow machine learning models on microcontrollers, allowing you to build AI/ML applications powered by deep learning and neural networks. With the included examples, you can recognize speech, detect people using a camera, and recognise "magic wand" gestures using an accelerometer. The examples work best with the Arduino Nano 33 BLE Sense board, which has a microphone and accelerometer.  
category=Data Processing  
url=https://www.tensorflow.org/lite/microcontrollers/overview  
ldflags=-lm  
includes=TensorFlowLite.h

This makes it visible in the Arduino IDE examples menu. Finally, navigate to the examples/ directory in the same place and rename the subdirectory to person\_detection. All these changes resolve the necessary differences to get the demo running.

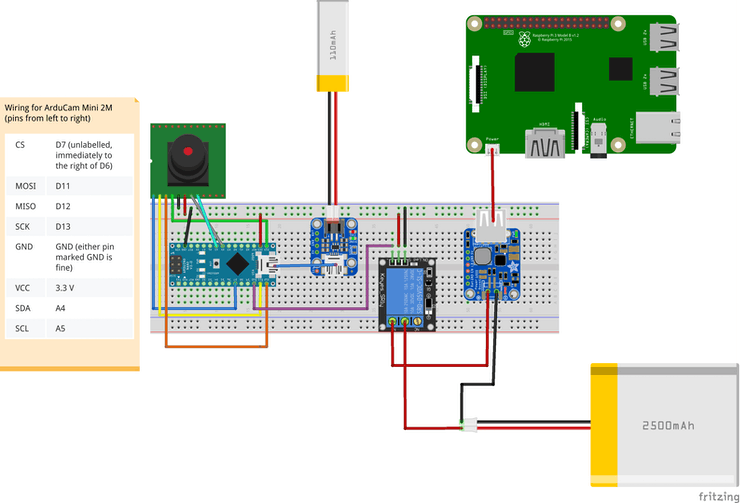
From here, you can follow the [official instructions](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro/examples/person_detection#install-libraries) to download the libraries for the ArduCam and JPEG decoding.

To run the example, navigate to Files -> Examples -> TensorflowLite:person\_detection and select the person\_detection sample script. Make sure you have your Arduino BLE 33 sense connected to your computer and it has been selected for flashing on the Arduino IDE. Flash the example and it should start up immediately. The on board LED will flash blue to indicate that an image has been captured and inference has been completed. The light will turn red if it did not detect a human and turn green when it has. Inference takes ~19 seconds since the person detection model is fairly large for the device.

**Using the Arduino as a Smart Switch**

Microcontrollers are very energy efficient, some being able to operate on coin cell batteries for years! With recent efforts to minimize machine learning models to run on embedded devices, we can build smart "switches" that can be invoked to power or trigger other devices that need more power to operate. With this person detection model running on the Arduino BLE sense, we can trigger a high energy consuming device like a Donkeycar using the [sombrero shield](https://store.donkeycar.com/products/sombrero). We'll use a relay connected to the Arduino and switch it on/off depending on inference results.

Wire up the relay, pi, and arduino following the wiring diagram below:



when person is detected, trigger the raspberry pi ON

Now simply add a few lines to the person\_detection.ino example to look at the inference results and switch the relay.

// at the top of the script, initialize a variable for the relay signal  
int relayOut = 10;  
  
// in the setup() loop, initialize the relayOut pin as output  
void setup(){  
 /\* .... \*/  
 pinMode(relayOut, OUTPUT);  
}  
  
// in the void() loop:  
void loop() {  
// Get image from provider.  
if (kTfLiteOk != GetImage(error\_reporter, kNumCols, kNumRows, kNumChannels,  
input->data.uint8)) {  
error\_reporter->Report("Image capture *failed*.");  
}  
// Run the model on this input and make sure it succeeds.  
if (kTfLiteOk != interpreter->Invoke()) {  
error\_reporter->Report("Invoke *failed*.");  
}  
TfLiteTensor\* output = interpreter->output(0);  
// Process the inference results.  
uint8\_t person\_score = output->data.uint8[kPersonIndex];  
uint8\_t no\_person\_score = output->data.uint8[kNotAPersonIndex];  
RespondToDetection(error\_reporter, person\_score, *no\_person\_score*);  
  
// Add the following lines:  
if (person\_score >= *no\_person\_score*){  
 digitalWrite(relayOut, HIGH);  
}  
else {  
 digitalWrite(relayOut, LOW);  
}  
  
}

**Extending our Smart Switch using BLE**

There may be situations where a high powered device could benefit from logic inferred elsewhere. For example, say you wanted a power hungry rover to start up when a person is detected in another section of a large warehouse. A way to extend the perception of our smart relay switch is through BLE.

BLE is a low power consuming wireless communication protocol designed to send small amounts of data. In this example, we set up an Arduino Nano 33 BLE Sense with the ArduCam attached to it as a control, sending a high alert when a person is detected. In the person\_detection script, along with other BLE initialization steps, we add the logic to send a high alert when the person\_score is high enough.

if (peripheral.connected()) {  
 if (sendsignal > 0){  
 alertLevelChar.writeValue((byte)0x02);  
 Serial.println("Wrote high alert");  
 }  
 else{  
 alertLevelChar.writeValue((byte)0x00);  
 Serial.println("Wrote low alert");  
 }  
}

A second Arduino Nano 33 BLE Sense will serve as a peripheral device that will listen for that alert and switch a relay attached to it.

while (central.connected()) {  
 //Serial.println("Getting Alert Level:");  
 if (alertLevelChar.written()){  
 if (alertLevelChar.value()){  
 digitalWrite(relay, HIGH);  
 Serial.println("Set relay to HIGH");  
 } else {  
 digitalWrite(relay, LOW);  
 Serial.println("Set relay to LOW");  
 }  
 }  
}