Bird Density, with Edge ML

by

Ryan Walker and Jin Seo

Bachelors of Technology in Electronics

Electrical and Computer Engineering Technology

British Columbia Institute of Technology

December - 2019

Bird Density, with Edge ML

by

Ryan Walker

Diploma in Robotics and Mechatronics, BCIT, 2011

Jin Seo

Jin put your degree here

Bachelors of Technology in Electronics

ELECTRICAL AND COMPUTER ENGINEERING TECHNOLOGY

We accept this report at conforming to the required standard

.....

.....

.....

.....

Bachelors of Technology in Electronics

Electrical and Computer Engineering Technology

British Columbia Institute of Technology

December - 2019

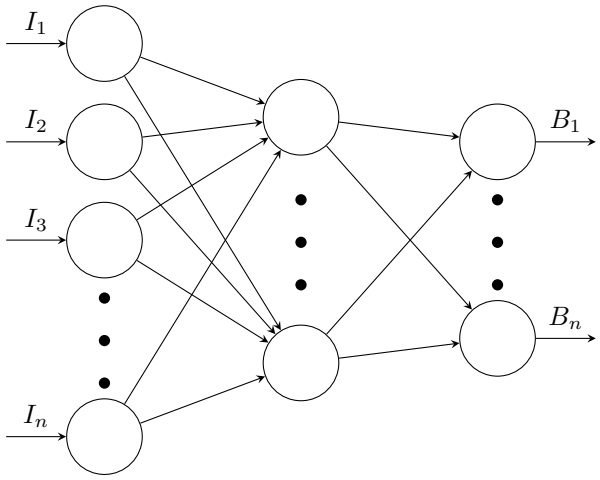
Contents

List of Figures

1 Introduction

Biologists have been aiming to understand bird migration for decades. Various custom hardware has been developed for doing this [1], however is it costly an requires capture and release of the bird. We are proposing the design, construction and deployment of a device that resides in a stationary location. This device is equip with a microphone and microcontroller, and by using machine learning techniques [2] [3] can detect and speciate various types of birds. In addition the device will also be able to record various types of environmental data for cross-correlation with bird density in post processing. Following the successful deployment we desire to make a bird density heat map of various regions and to undestand how it changes over time.

2 Machine Learning on the Edge



Machine learning and AI have been popular topics in the last couple of years, especially the use of neural networks in classification algorithms. A neural network is a collection of artificial neurons which can loosely model the neurons in a biological brain. They are connected is a way that allows data to propagate through them, each neuron connection (edge) has a weight associated with it, this weight either increases or decreases the strength of the signal at a connection. The neurons are arranged in "layers", the first and last layers are known as the input and output layers respectively, and the middle layers are known at the hidden layers. "Training" a neural network is the automated process of adjusting the weights on a sample dataset so that the output gives the requred value for a given input. As an example, if were given a problem to classify 1000 pictures into either "hotdog" or "not hotdog", where the images were 250 pixels by 250 pixels. You would start by:

- Manually label a sufficiently large section of the dataset (say 500 pictures)
- Introduce 62500 input neurons ($250 \cdot 250$)
- Introduce 1 output neurons (Hotdog or not hotdog)
- Train the network so output says hotdog for all the hotdog pictures, and not hotdog for all the not hotdog pictures
- When you then feed in puctures it hasn't seen before, it should classify the remaining data correctly

The accuracy is given by the ratio of correct classifications to incorrect classifications. It has not been until recently that these applications have moved into edge compute, meaning all the processing is run on the device as apposed to cloud. We propose to design a NN (Neural Network) that can be run on a STM32 microcontroller for classification of various birds. As this is such a rapidly expanding field there are various software implementations developed for embeded applications [4] [5].

3 Signal processing

This project is required to utilize several signal processing techniques. Some of these techniques could be included filters, discrete Fourier transforms, and so on. Filters are necessary to improve the quality of the bird calls and remove unwanted noises. In addition, in order to come up with digitized data from acoustic analog signals, other techniques could be applied such as Discrete wavelet transform or mel-frequency cepstrum. In this case, MATLAB can be used for simulation.

3.1 Mel Spaced Filter Bank

Human hearing sensitivity works in a very nonlinear way. For a human, there is a stark difference between a signal that is 50hz and a signal that is 150hz. However is it almost impossible to notice a difference between a a 10khz signal and a 11khz signal. The mel scale has been developed to represent this. It it shown in Equation 1 and is plotted in Figure 1.

$$M(f) = 1125 \cdot log(1 + \frac{f}{700}) \tag{1}$$

To get more adequate feature extraction we are going to use a mel spaced filter bank. This effectively maximizes the features extracted as a function of the mel scale. The filter array is shown in Figure 2. In this example we have implemented ten filters for visulizing purposes, however in practice we are going to implement over twenty.

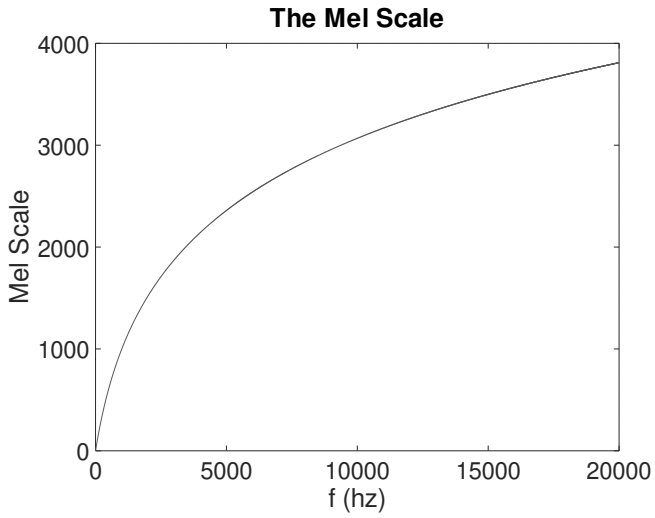


Figure 1: The Mel Scale

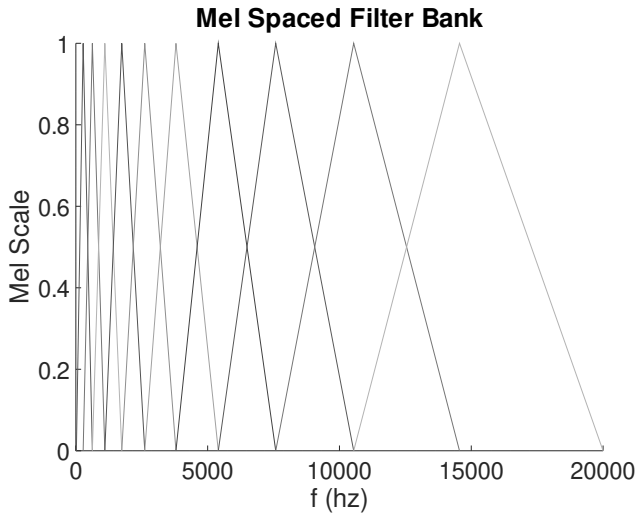


Figure 2: The Mel Space Frequency Array

4 Software

In the project we are planning on using Matlab/Simulink for any filter design, C/C++ for embedded code and possibly some python for post processing.

5 Hardware

The hardware will need to be battery powered in order to collect data over extended periods of time. We are currently targeting the specifications below.

Spec	Metric
Battery Life	1 week
Compatable Birds	10
Cost (QTD 100)	$< \$30$

It should be inexpensive so it becomes feasible to deploy many devices and should be completely stand alone. In addition it should have a wiresless uplink/downlink to transmit data to and from a computer.

5.1 Microcontroller

In addition to having experience with the STM32 family of microcontrollers, TensorFlow Embedded [5] has extensive documentation for implementation on the STM32F746 [6]. This part has 1MB flash, 320k RAM and runs at 216Mhz. TensorFlow embedded is designed to run on 20k of flash, so we will have more than enough space for the NN and application code.

5.2 Microphone

The device will contain a microphone that will be used to record ambient noises. A suitable microphone has not yet been selected.

5.3 Architecture

- **MCU:** STM32F746.
- **SD card:** For recording data.
- **18650 Cell:** 2x 18654 Li-ion cells for power.
- **Microphone:** Microphone for recording sound.
- **Temperature Sensor:** Temperature sensor for recording ambient temperature.

6 Data Harvesting and Conditioning

In order to train the neural net, we require a great deal of data. This was gatherd from a website where bird listeners post sounds of birds [7]. To this day we have gathered and labeled 589 crow samples, of which roughly half are not crows and 691 sanpiper samples to which roughly half are not sanpipers. We did this using the workflow outlined below.

6.1 Harvesting

The script below harvest the data from a given bird.

```
#!/bin/bash

Name='sandpiper' # Bird Name
Page=2 # Start at the first page

wget "https://www.xeno-canto.org/explore?query=${Name}&pg=${Page}"

fname="explore?query=${Name}&pg=${Page}"
cat ${fname} | grep download | grep -o -P "href.*download'" | tr -d "href='" > out
rm ${fname}

mkdir $Page
mv out $Page
cd $Page

for i in {1..30}
do
  dl="$(sed -n "${i}p" out)"
  echo $dl
  wget "https://www.xeno-canto.org${dl}"
done

rm out

cd ..
```

6.2 Labeling

The labeling process was done manually, it was done with the script below essentually this script will hunt through a file and find sound files, then it will chunk off a 1s piece and play it, if the user presses 'y' then it will save the file down as "n_crow" if the user pressed 'n' then it will save the file down a "n_Notcrow", where n is an incremented integer.

```
# python splice.py <infolder> <outfolder> <birdName> <StartingFileNumber>
import os
import sys
import pdb
from os import listdir
from os.path import isfile, join
from getch import getch

gCount = 0                                                    # Global Counter
```

```
infolder = sys.argv[1]                                    # Folder name to go into
outfolder = sys.argv[2]
birdName = sys.argv[3]
startingPoint = int(sys.argv[4])

files = [f for f in listdir(infolder) if isfile(join(infolder, f))]
os.system("mkdir " + outfolder)

from pydub import AudioSegment
from pydub.playback import play

numFiles = len(files)

for p in range(0, numFiles):
  print('Opening file: %s' % files[p])
  sound = AudioSegment.from_file(infolder + '/' + files[p])
  if sound.frame_rate == 44100:
    length = int(len(sound)/1000)

    for k in range(1, length):
      chunk = sound[(k-1)*1e3:k*1e3]
      if gCount > startingPoint:
        play(chunk)
        print(files[p]+':'+ str(gCount) + ", y/n???")
        ans = getch()
        if ans == 'y':
          out_f = open(outfolder + "/%d_%s.mp3" % (gCount, birdName), 'wb')
          chunk.export(out_f, format='mp3')
        else:
          out_f = open(outfolder + "/%d_Not%s.mp3" % (gCount,birdName), 'wb')
          chunk.export(out_f, format='mp3')
      gCount+=1
```

6.3 Conditioning

Our neural network is deigned to accepts frequency data, not sound data. So there is some conditioning required beforehand. This script opens up the output files from the last script and preforms a FFT. It then reduces the number of bins to something more manageable, right now we are using 128 input neurons, so there are 128 fft bins. It then saves all this to a single json file.

```
import matplotlib.pyplot as plt
from os import listdir
from os.path import isfile, join
from scipy.io import wavfile as wav
from scipy.fftpack import fft, fftfreq, fftshift
```

```
import numpy as np
import json
import pdb
from python_speech_features import mfcc
from python_speech_features import delta
from python_speech_features import logfbank
from python_speech_features import ssc
import librosa

folders = ['crow', 'sandpiper']
jsonFname = 'data.json'
bins = 128
root = []

for birds in range(0, len(folders)):
  files = [f for f in listdir(folders[birds]) if isfile(join(folders[birds], f))]
  for l in range(0, len(files)):
    fname = files[l]
    fs, data = wav.read(folders[birds]+'/'+fname)   # Open the file
    if(fs == 44100):
      try:
        data = data[:,0]                                  # Remove one of the tracks (convert to
            mono)
      except:
        continue

      mfcc_feat = mfcc(data, fs, nfft=1103).tolist()
      # fbank_feat = logfbank(data, fs)

      if "Not" in fname:
        isBird = False
      else:
        isBird = True

      data = []
      data = mfcc_feat

      root.append({
        'fname'    : fname,
        'fft'      : data,
        'bird'     : folders[birds],
        'isbird'   : isBird,
        'fs'       : fs
      })

with open(jsonFname, 'w') as outfile:
  json.dump(root, outfile, indent=2, sort_keys=True)
```

7 The Neural Network

7.1 Background and Exsisting work

The idea of Artificial Neural Networks comes from the structure and functions of a human brain. Neurons in human brain that process and transmit information between themselves.

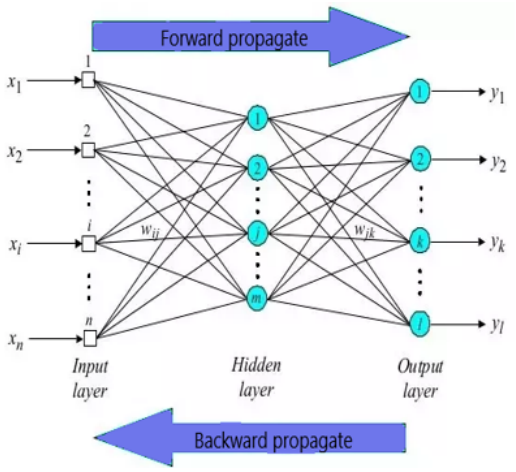


Figure 3: Forward&Backward Neural Network

7.1.1 Perceptron

People used perceptron to model of decision-making. Perceptron works like step function. So, if perceptron is over a certain threshold, it outputs 1 otherwise 0. Perceptrons act as a linear decision; however, many reality functions are NOT linear.

$$Output = \begin{cases} 0, & if\ w \cdot x + b \leq 0 \\ 1, & if\ w \cdot x + b > 0 \end{cases}$$

Figure 4: equation of perceptron.

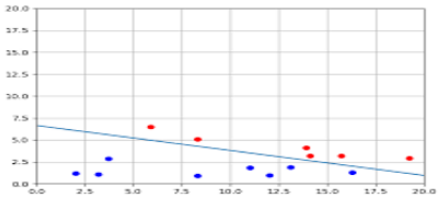


Figure 5: linearity of perceptron.[9]

7.1.2 Sigmoid

Sigmoid neuron has weights for each input w1,w2,..., and overall bias b. Its $\sigma(w \cdot x + b)$ which is called the sigmoid function So, the sigmoid can be expressed as:

$\sigma(z) = \frac{1}{1+\exp(-z)} \ldots (3)$

We chose Tensorflow for out neural network design because one you train up a model, you have the ability to convert it to a "tflite" model which can be run on a microcontroller using tensorflow lite. We then built the model structure and trained the model. Our current model is shown in Figure 6, it has 128 input neurons on the input layer (for our 128 fft bins) 128 neurons on the one hidden layer and 1 neuron on the out (either crow or not crow). The script starts by randomly splitting our dataset into the training and validation set, where the training set it used for the training while the validation set is used for validation after the training is complete, that is to say the network has no knowledge of the validation set when adjusting the weights. After this process our network got **93% accuracy on the validation set**

```
# https://towardsdatascience.com/getting-started-on-deep-learning-for-audio-data-667d9aa76a33
# https://www.tensorflow.org/lite/convert/python_api

import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import pdb
from tqdm import tqdm
from sklearn.model_selection import KFold, cross_val_score, train_test_split

plt.style.use('fivethirtyeight')
print(os.listdir("../input"))

# train = pd.read_json('../input/train.json')
train = pd.read_json('../../data/data.json')

train_train, train_val = train_test_split(train, random_state = 42) # Split into training set and
    validation set

xtrain = [k for k in train_train['fft']]
ytrain = (train_train['bird'].values == 'crow') & (train_train['isbird'].values)
xval = [k for k in train_val['fft']]
yval = (train_val['bird'].values == 'crow') & (train_val['isbird'].values)

# Pad the audio features so that all are "10 seconds" long

x_train = tf.keras.preprocessing.sequence.pad_sequences(xtrain)
x_val = tf.keras.preprocessing.sequence.pad_sequences(xval)
y_train = np.asarray(ytrain)
y_val = np.asarray(yval)

model = tf.keras.Sequential()
```

```
# model.add(tf.keras.layers.Dense(1,input_shape=(128,)))
x=len(xtrain)
y=len(xtrain[0])
z=len(xtrain[0][0])

model.add(tf.keras.layers.Dense(1,input_shape=(y,z)))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer = tf.keras.optimizers.Nadam(lr=0.001), metrics
    =['accuracy'])
print(model.summary())
tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True)

# fit on a portion of the training data, and validate on the rest
# from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_acc', factor=0.1, patience=2,
    verbose=1, min_lr=1e-8)
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', verbose=1, patience=20,
    restore_best_weights=True)
history = model.fit(x_train, y_train,batch_size=512, epochs=1000, validation_data=[x_val, y_val],
    verbose = 2,callbacks=[reduce_lr,early_stop])

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

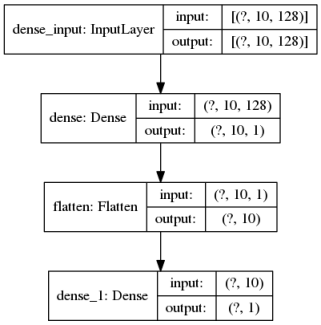


Figure 6: Current Model

8 Hardware

The first revision of the hardware is finished and we have connected to the microcontroller, we have blinked on of the LEDs but have not implemented any sampling or the neural network. The physical hardware is shown in Figure 7 while the schematic is shown in Figure 8 and 9. We got a PCB manufactued overseas ans placed all the SMD components onto it.

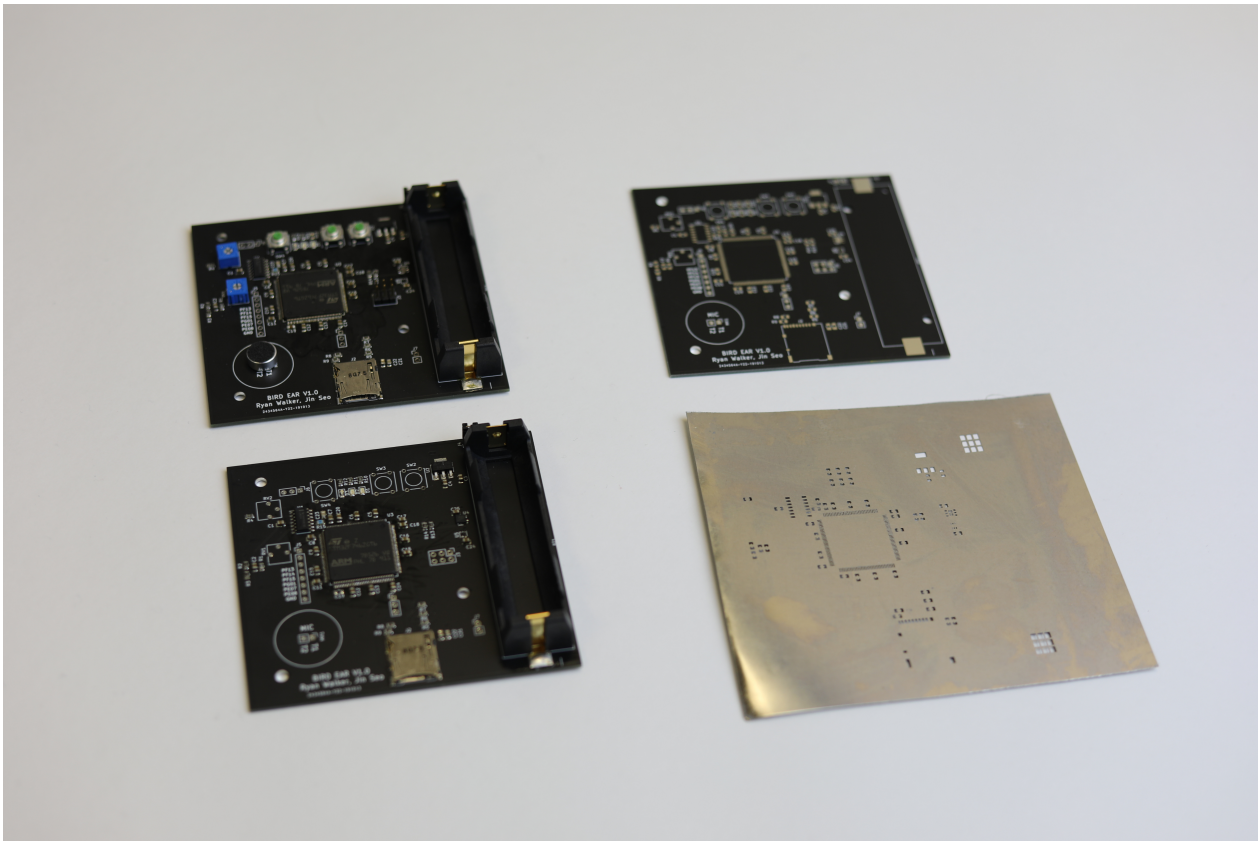


Figure 7: Hardware

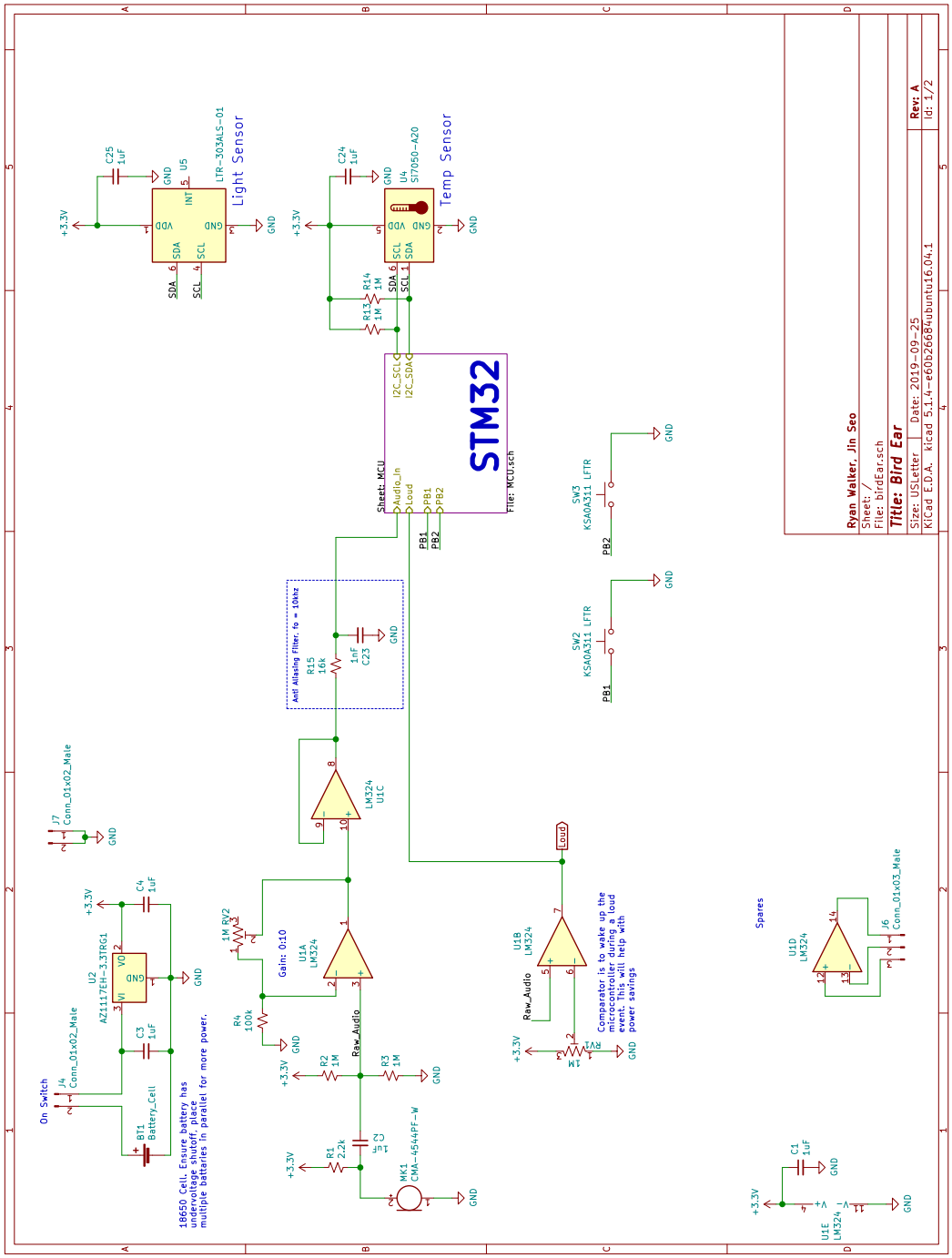


Figure 8: Schematic - page 1

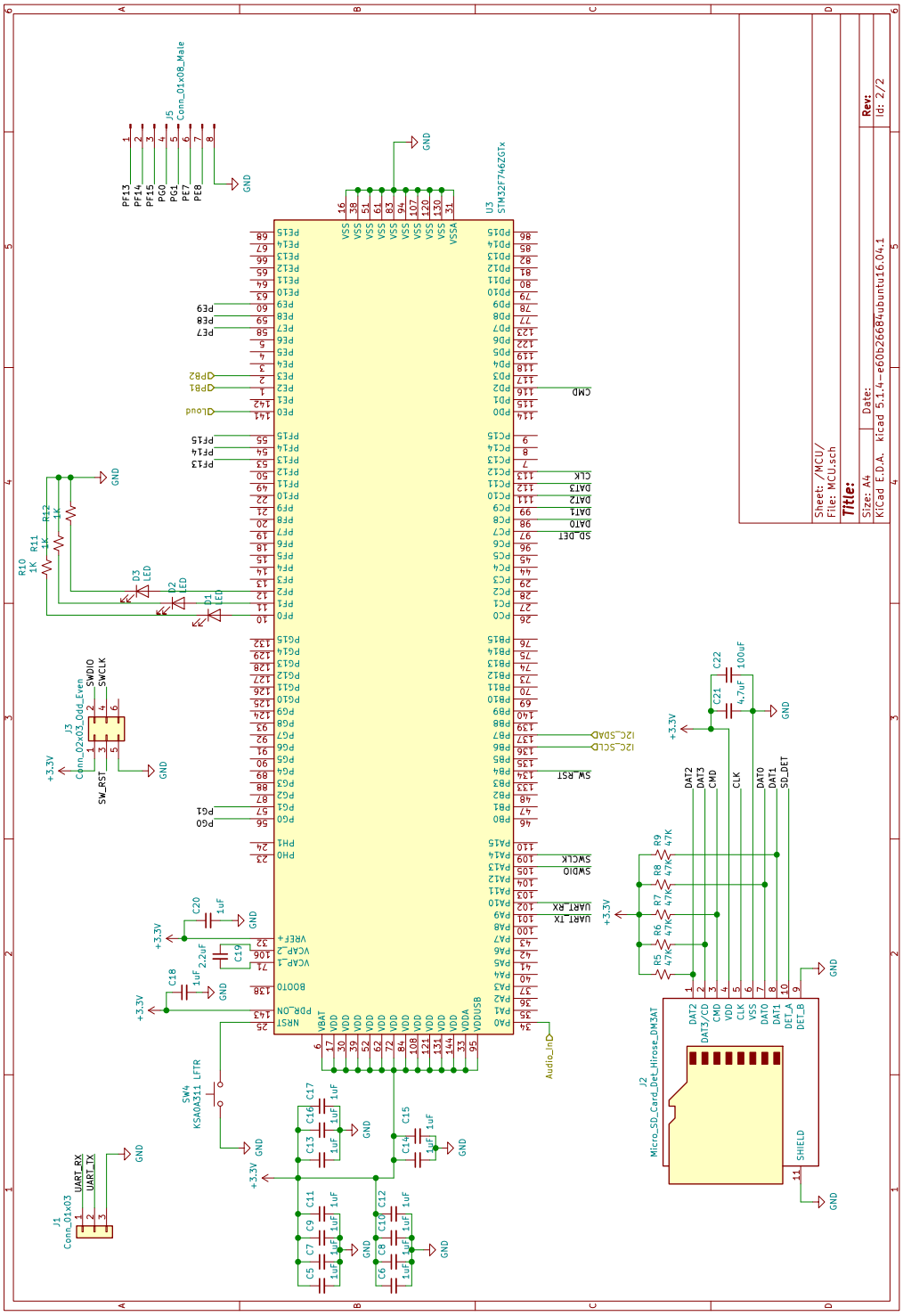


Figure 9: Schematic - page 2

9 Next Steps

The next steps of the project are...

- Implement mel-spaced frequency array.
- Increase the number output neurons to 2, so we can classify between crows and sandpipers.
- Add another three types of birds to the dataset.
- Research "VGG like feature extraction".
- Implement microcontroller firmware for audio sampling.
- Implement neural network on the microcontroller.
- Deploy the device in the field.

References

[1] Bird Tracking Hardware. `https://atstrack.com/animal-class/avian.aspx` [Accessed June 22, 2019]

[2] Stowell, Wood, Pamua, Stylianou4, Glotin "Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge" `https://arxiv.org/pdf/1807.05812.pdf`

[3] Ilyas Potamitis, "Deep learning for detection of bird vocalisations" `https://arxiv.org/pdf/1609.08408.pdf`

[4] CMSIS NN Software Library https://www.keil.com/pack/doc/CMSIS/NN/html/index.html

[5] TensorFlow Lite for Microcontrollers https://www.tensorflow.org/lite/microcontrollers/overview

[6] ARM Cortex-M7 STM32F7 Microcontroller IC 32-Bit 216MHz 1MB (1M x 8) FLASH 100-LQFP (14x14) https://www.digikey.ca/product-detail/en/stmicroelectronics/STM32F746VGT6/497-15819-ND/5287178

[7] xeno-canto is a website dedicated to sharing bird sounds from all over the world. https://www.xeno-canto.org/

[8] Bird Call Identifier. `https://web.wpi.edu/Pubs/E-project/Available/E-project-042910-001603/unrestricted/Bird_Call_Identification_MQP_2010.pdf` [Accessed April 29, 2010]

[9] Calculate the Decision Boundary of a Single Perceptron https://medium.com/@thomascountz/calculate-the-decision-boundary-of-a-single-perceptron-visualizing-linear-separability-c4d77099ef38

[10] Neural Networks and Deep Learning http://neuralnetworksanddeeplearning.com/index.html [Accessed Determination Press, June, 2019]