Final Report

Arduino project group 8
By: Tijn Hassing 3004619, Alexandru Lungu 3006301, Narendra Setty 2944200, Hoang Pham - 3002152 & Joris Faas 2985888

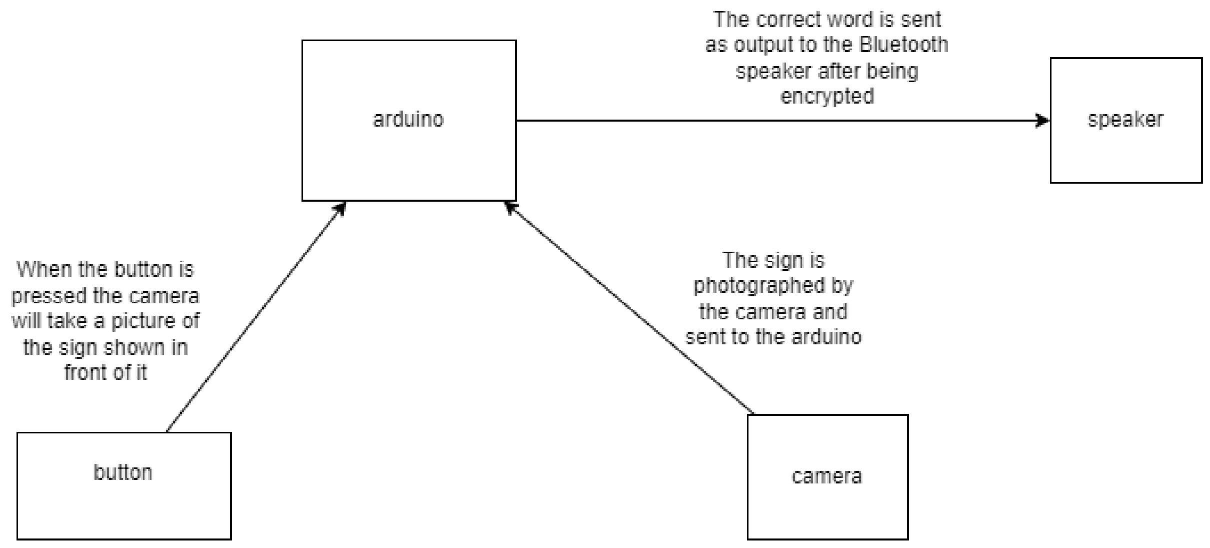
Introduction **1**
Functional Details **2**
Block diagram 2
Sequence Diagram 2
Technical details and explanation **3**
Machine learning implementation 3
Few shot 3
Lenet5 3
Arduino implementation 3
Text-to-speech 3
uploading the model 3
Implementation results 4
Conclusions and future directions 5
Individual contributions and proposal for grading correction 6

Introduction

Our project is called SLAIT, the Sign Language Arduino Interpreter with Text-to-Speech. It is a system used to be able to read signs and convert them to spoken words. This works by making pictures of each sign used to spell out the word, converting each sign to a letter using our lenet5 model and then sending it via bluetooth to a text-to-speech device. This way it can be used by people unfamiliar with sign language to still be able to converse with them. This way all people regardless of sensory abilities can engage in meaningful interactions with each other.

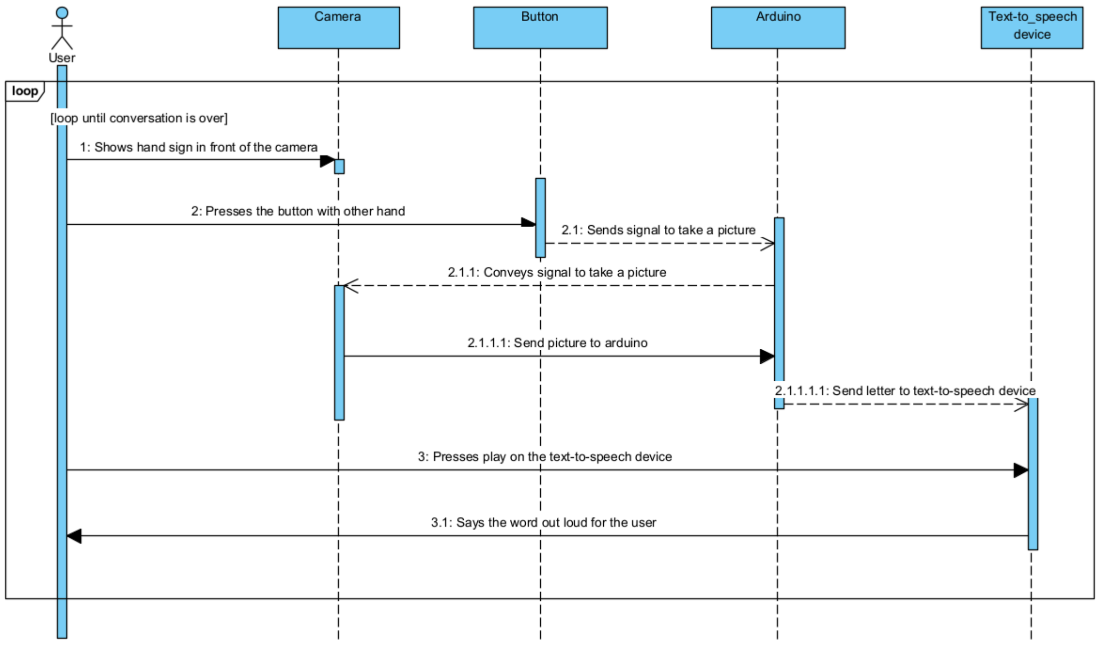
Functional Details

Block diagram


arduino
The correct word is sent as output to the Bluetooth speaker after being encrypted
speaker
When the button is pressed the camera will take a picture of the sign shown in front of it
The sign is photographed by the camera and sent to the arduino
button
camera

Our system consists of 4 main elements working together with the arduino as the central point.

Sequence Diagram


User
Camera
Button
Arduino
Text-to_speech device
loop
[loop until conversation is over]
1: Shows hand sign in front of the camera
2: Presses the button with other hand
2.1: Sends signal to take a picture
2.1.1: Conveys signal to take a picture
2.1.1.1: Send picture to arduino
2.1.1.1.1: Send letter to text-to-speech device
3: Presses play on the text-to-speech device
3.1: Says the word out loud for the user

Initially the user places their hand facing the camera and presses the button with their free hand. The button which is on the arduino sends a signal to the arduino to take a picture and the arduino in turn sends a signal to the camera to take a picture. Subsequently, the arduino sends the picture to the arduino where the machine learning model on the arduino converts the image into its appropriate letter. Following this, the arduino sends the letter to the text-to-speech device. When the user has completed a word they can access that word by pressing play on the text-to-speech device which then repeats the letter.

Requirements

User Interface

1.1 The system shall have a button that the user can press with their free hand to initiate the image capture process.

1.2 When the user places their hand facing the camera and presses the button, the system shall send a signal to the Arduino to take a picture.

Image Capture and Processing

2.1 When the camera takes the picture, the Arduino shall process the image using the machine learning model to convert the image into its appropriate letter.

Machine Learning and Conversion Requirements

3.1 The machine learning model on the Arduino shall convert the captured image into a corresponding letter.

3.2 When an image is successfully captured and processed, the Arduino shall generate the corresponding letter based on the image.

Communication and Data Transmission Requirements

4.1 When the image is converted into a letter, the Arduino shall send the letter to the text-to-speech device.

Text-to-Speech Device Requirements

5.1 The text-to-speech device shall store the received letters sequentially to form a word.

5.2 When the user presses the play button on the text-to-speech device, it shall repeat the formed word.

Technical details and explanation

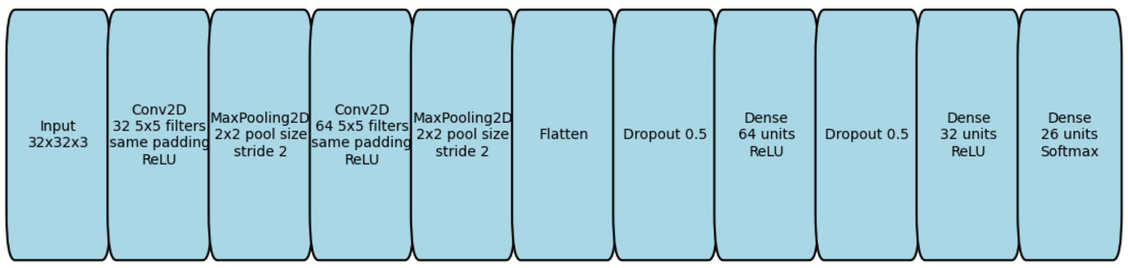
Machine learning implementation

Lenet5

Our Lenet5 model makes use of 5 layers in order to be small enough to be placed on the arduino. As input it takes images of shape (32, 32, 3). Our model loads a labeled dataset of 1656 images, divides this into train and validation samples.

Below an image of the layers in our model is given, we use 2 conv layers and 3 dense layers to process the result with some layers of max pooling, to reduce the size of the input, flatten, to convert it to a 1 dimensional vector and dropout, to prevent overfitting. It ends with 26 different classes, one for each letter.

LeNet-5 Model Architecture



We furthermore make use of an Adam optimizer with a learning rate of 0.0001, a batch-size of 16 and 1000 epochs.

Few shot

The code copies the weights of a pre-trained CNN model and then uses them to create our own pre-trained model for few-shot learning. This is used when only a few examples per class are available for training. Initially the dataset is loaded and pre-processes to make it compatible with the model. The few-shot learning task is created by randomly selecting the required number of classes and then generating a larger training set and a smaller query set (for testing) by dividing the dataset.
The evaluation involves multiple iterations of creating and assessing the few-shot tasks where the model predicts the class of each query image based on the closest match to the support images mean (prototype). The accuracy is then computed by comparing the predicted labels with the actual labels of the images.

Arduino implementation

uploading and pruning the model

The model that we have put on the arduino has first been trained, then it was pruned and fine tuned to get the size small enough to fit the model on the arduino. After pruning the model gets quantized and converted into a tflite file. Once this file is obtained you can use the unix command “xxd -i model_quantized2.tflite > model_data.cc” to convert the tflite file to a C array. After this conversion we only had to add the C array file to our arduino sketch and that is how we uploaded our trained model onto the arduino.

Text-to-speech

Our arduino code has a couple of different components, the first one being the bluetooth connection part to send characters to the text-to-speech app on my phone. Here we use the ArduinoBLE library to build a BLE HID (Bluetooth Low energy Human Interface Device) to essentially mimic a bluetooth keyboard that sends characters over the connection.

The second part is the camera activation. This is code that was provided for us to use during the ICES labs. We used this code with some minor tweaks like converting all images to grayscale to set up our camera.
The last and most important part of our application is the model (explained above how to upload onto arduino). When a picture is taken it gets converted to grayscale, interpreted by the trained model and the predicted letter will then be displayed and also sent over the bluetooth connection.

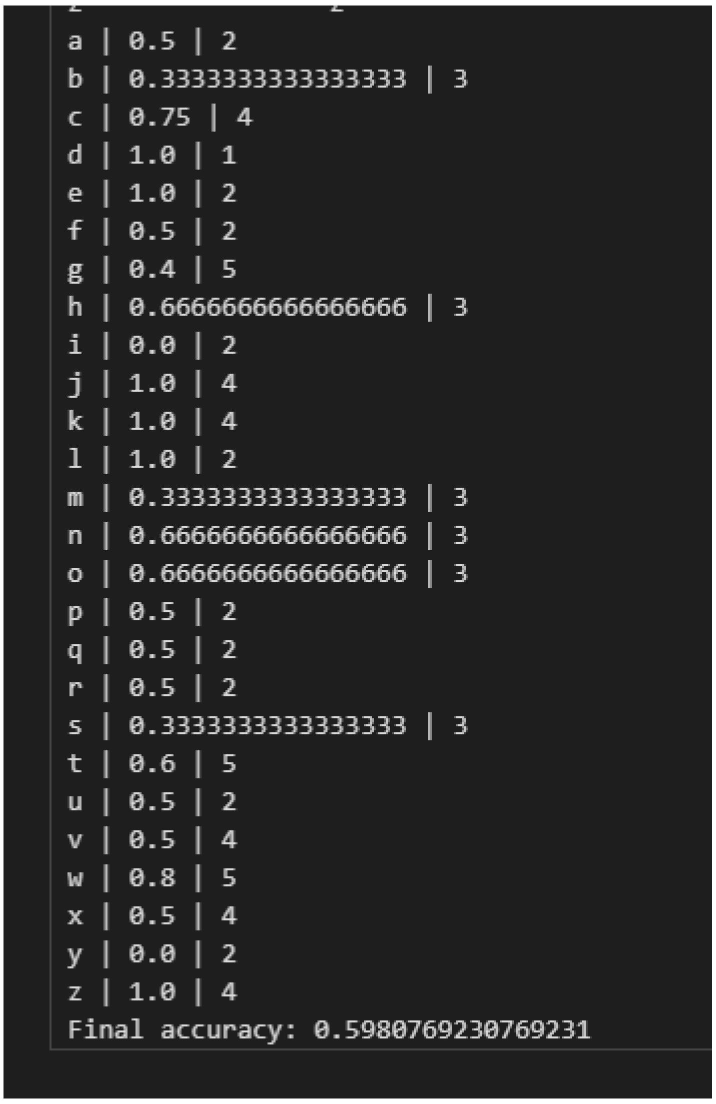
Implementation results

Machine learning implementation

Lenet5

The accuracies displayed below show the results for each letter and also the number of images we had in the dataset for them. There appears that some letters have very bad predictions, while for others the model is able to recognize them frequently, for example letters like ‘z’, ’k’ or ’r’. Our test set is also a bit unbalanced regarding the number of images per letter therefore, this generates some very low or very high accuracies for specific characters. In the end we get a final accuracy of 60%.

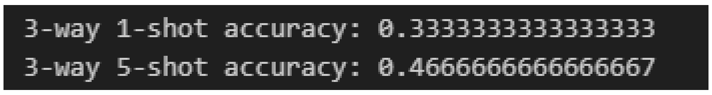
Actual letter | accuracy | number of letters in the test set



```
a | 0.5 | 2
b | 0.3333333333333333 | 3
c | 0.75 | 4
d | 1.0 | 1
e | 1.0 | 2
f | 0.5 | 2
g | 0.4 | 5
h | 0.6666666666666666 | 3
i | 0.0 | 2
j | 1.0 | 4
k | 1.0 | 4
l | 1.0 | 2
m | 0.3333333333333333 | 3
n | 0.6666666666666666 | 3
o | 0.6666666666666666 | 3
p | 0.5 | 2
q | 0.5 | 2
r | 0.5 | 2
s | 0.3333333333333333 | 3
t | 0.6 | 5
u | 0.5 | 2
v | 0.5 | 4
w | 0.8 | 5
x | 0.5 | 4
y | 0.0 | 2
z | 1.0 | 4
Final accuracy: 0.5980769230769231
```

Few shot

The result for few shot is not always as expected, sometimes it provides with better accuracy for 1-shot, but in most cases the accuracies are as shown below. Few shot algorithm uses a set of 5 folders each containing 10 images taken by us, a folder represents a letter.



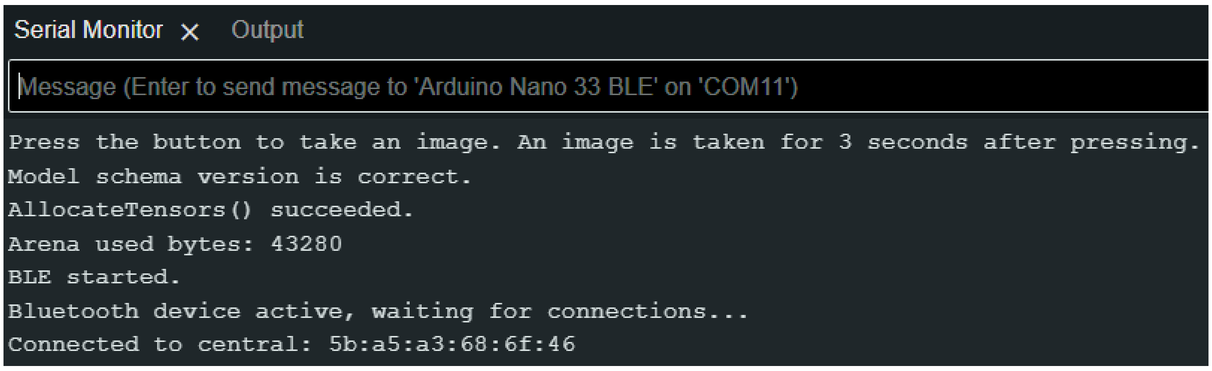
```
3-way 1-shot accuracy: 0.3333333333333333
3-way 5-shot accuracy: 0.4666666666666667
```

Arduino implementation

uploading and pruning the model

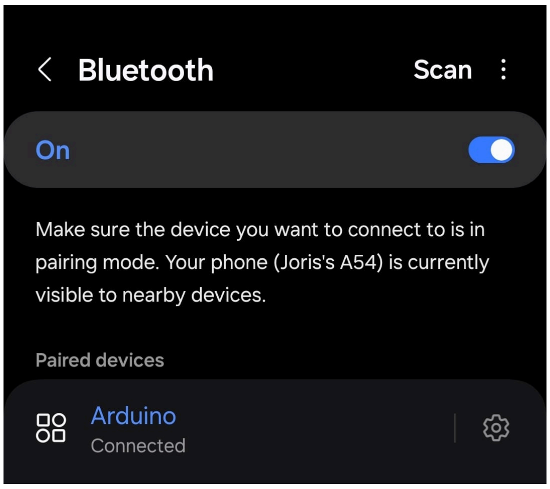
The result of our saving of the model, loading the model and pruning/quantizing the model can be seen in the folder provided together with this report. The model is made significantly smaller than it would have been otherwise, which makes it able to run on the arduino.

Text-to-speech


Serial Monitor
Output
Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM11')

```
Press the button to take an image. An image is taken for 3 seconds after pressing.
Model schema version is correct.
AllocateTensors() succeeded.
Arena used bytes: 43280
BLE started.
Bluetooth device active, waiting for connections...
Connected to central: 5b:a5:a3:68:6f:46
```

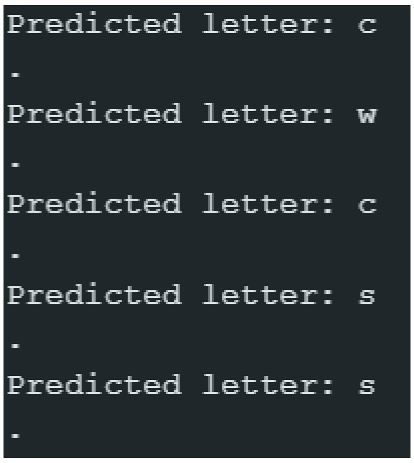
As you can see on the corresponding images, the arduino is able to connect to my phone with the help of bluetooth where the device will first show up under the name “Nano 33 BLE HID Keyboard”. After you choose to pair with this device it will show up under paired devices with the name “Arduino”, when you see this device connected, the setup has been successful. We tested this feature by disconnecting and reconnecting both the arduino and the other bluetooth device multiple times to see if it would still connect, which it did.



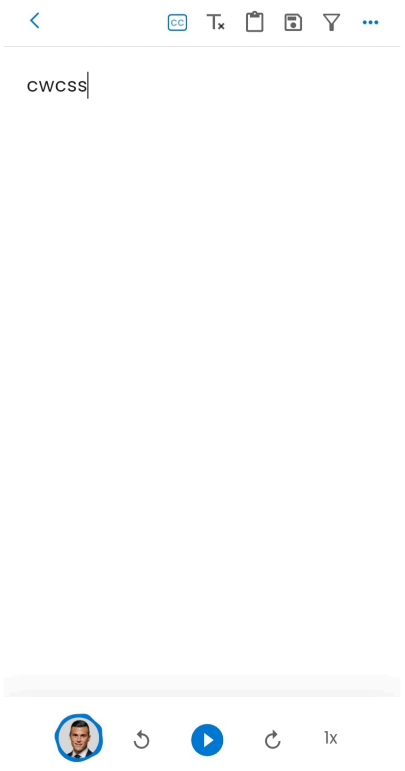
Make sure the device you want to connect to is in pairing mode. Your phone (Joris's A54) is currently visible to nearby devices.

As the screenshots to the right show, the model predicts which letter is being signed and then sends the predicted letter to a bluetooth device that has a text to speech app open.

The only problem with our implementation is that the predictions are not very accurate. There are multiple factors that may have caused this. First of all, trying to shrink the model to actually fit on the arduino has impacted the accuracy of the predictions.



```
Predicted letter: c
.
Predicted letter: w
.
Predicted letter: c
.
Predicted letter: s
.
Predicted letter: s
.
```



cwcss

Secondly, the images that are being interpreted might be too small for the model to make out the defining features of all letters.

And lastly, during the entire project we have only had a working arduino for 2 days due to the ICES assignment and our first arduino not functioning properly. So we had very little time to bugfix our application which in our opinion is the biggest reason for our lack of accuracy.

Conclusions and future directions

This project's result is not as good as it could have been, we had a broken arduino for most of the time, but we were also stuck on our Machine Learning model for quite a while since we used an unsuitable dataset. Had we figured this out earlier or with a little bit more time we could have had a significantly better accuracy. We are satisfied with our results however, since we have 26 different classes our accuracy is pretty good. We have been able to implement everything we wanted onto the arduino even though we haven’t had enough time to test and improve our accuracy on the deployed model.

So in general we all learned a lot from the project and for our first Machine Learning model it performs adequately.

For the future, we would like to have planned the project and the division of work more efficiently. We would also have liked to have explored a larger variety of datasets as we had used a single dataset for much of the initial part of the project. We also wanted to have tested with more types of augmentations and also tested with different types of structures of models.

Individual contributions and proposal for *grading correction*

	Joris	Tijn	Alexandru	Narendra	Hoang
Arduino bluetooth	X				
LeNet model		X			
Few Shot model			X	X	
pruning & quantizing	X	X			
Arduino model integration	X				
Report	X	X	X	X	

For the grading correction, if we have a passing grade, we would like to keep Hoang's grade at a 5.5 and distribute all the extra points over all other group members if this is possible so everyone still has a passing grade. If this is not possible we would like to all receive the same grade.