IOT based Baby Monitoring System

Problem Statement

Child care is necessary for parents, but nowadays taking care of a child has become a lot more challenging, especially for working mothers. It has become increasingly difficult for parents to continuously monitor their baby’s condition. Thus, a smart baby monitoring system based on IoT and machine learning is implemented to overcome the monitoring issues and provide intimation to parents in real-time.

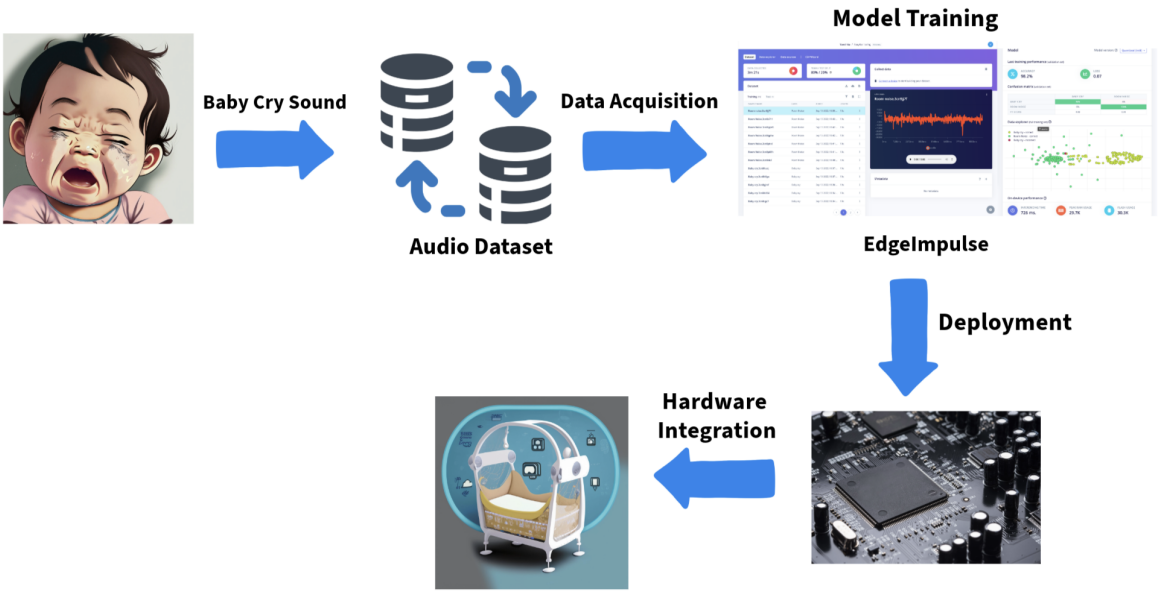
The current design of baby swing rockers, featuring only a manual ON/OFF switch and a swing timer option, poses a significant challenge for parents seeking to soothe a crying infant during nighttime hours. The inherent limitation of the rocker being in the "Off" position by default hinders its immediate responsiveness, leaving parents with a less effective tool for calming a distressed baby during crucial moments.

This limitation calls for an innovative solution that addresses the need for enhanced functionality and accessibility in baby swing rockers, particularly during nighttime use, to provide a more responsive and reliable soothing experience for both infants and their caregivers.

Software Front

We've implemented a TinyML model to automate the movement of the baby swing. In order to calm the infant and aid in their return to sleep, an Arduino, equipped with audio inferencing capabilities, is employed to differentiate the sound of the crying infant from ambient noise and activate the baby swing rocker motor.

Architecture



The project's general architecture is explained in the block diagram above. Following training within the Edge Impulse platform, the TinyML model is subsequently redeployed onto an Arduino Uno.
In order to train the TinyML model with enough data to yield acceptable accuracy, We have gathered background room noise and infant crying sounds. Lastly, we have integrated the system with the swing hardware.

Data Acquisition

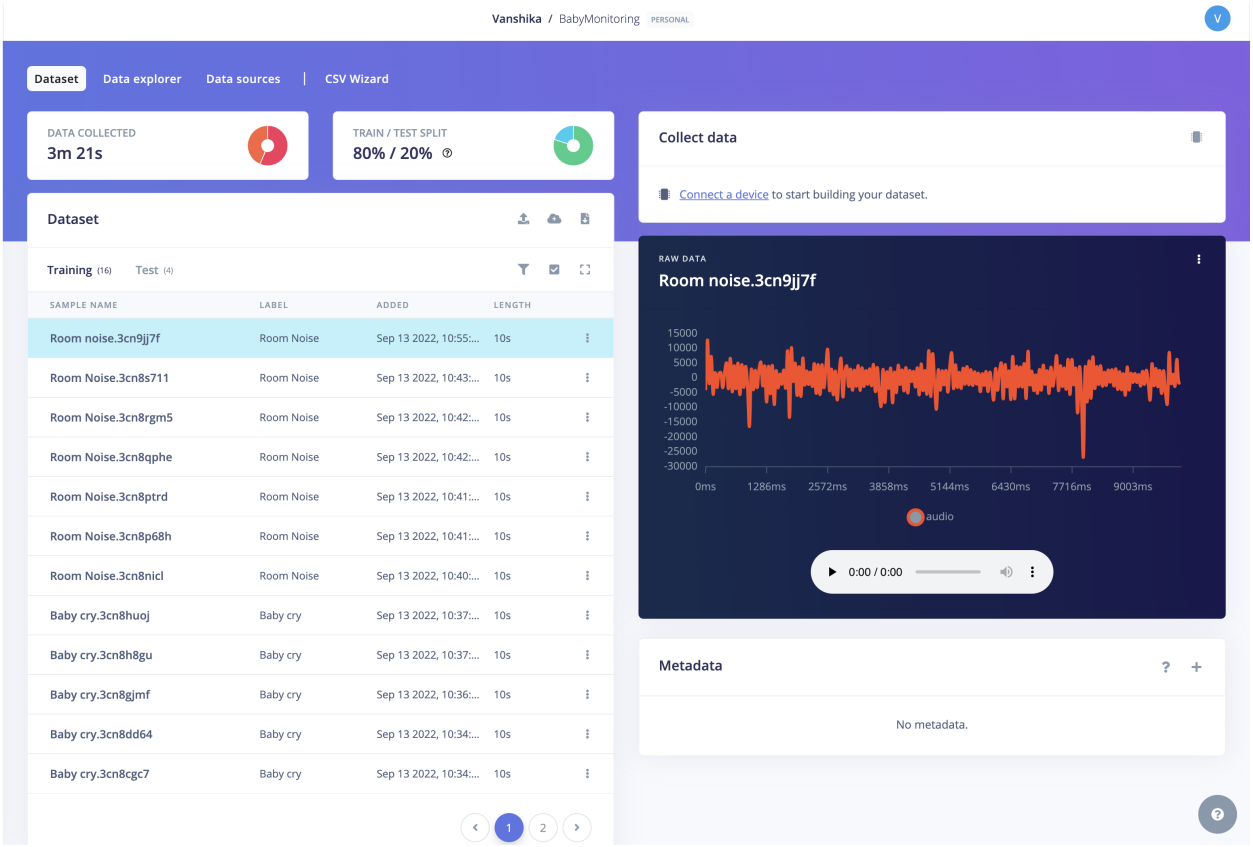
We have recorded actual infant cries and background noise to gather statistics. To gather audio samples, We have used an Arduino UNO equipped with a Vision Shield. The reason the Arduino UNO is used is that it has two 16 MHz MP34DT05 microphones.

After the Arduino UNO hardware has been flashed with the UNO firmware, launch your system's command window and enter the following command:

```
edge-impulse-daemon
```

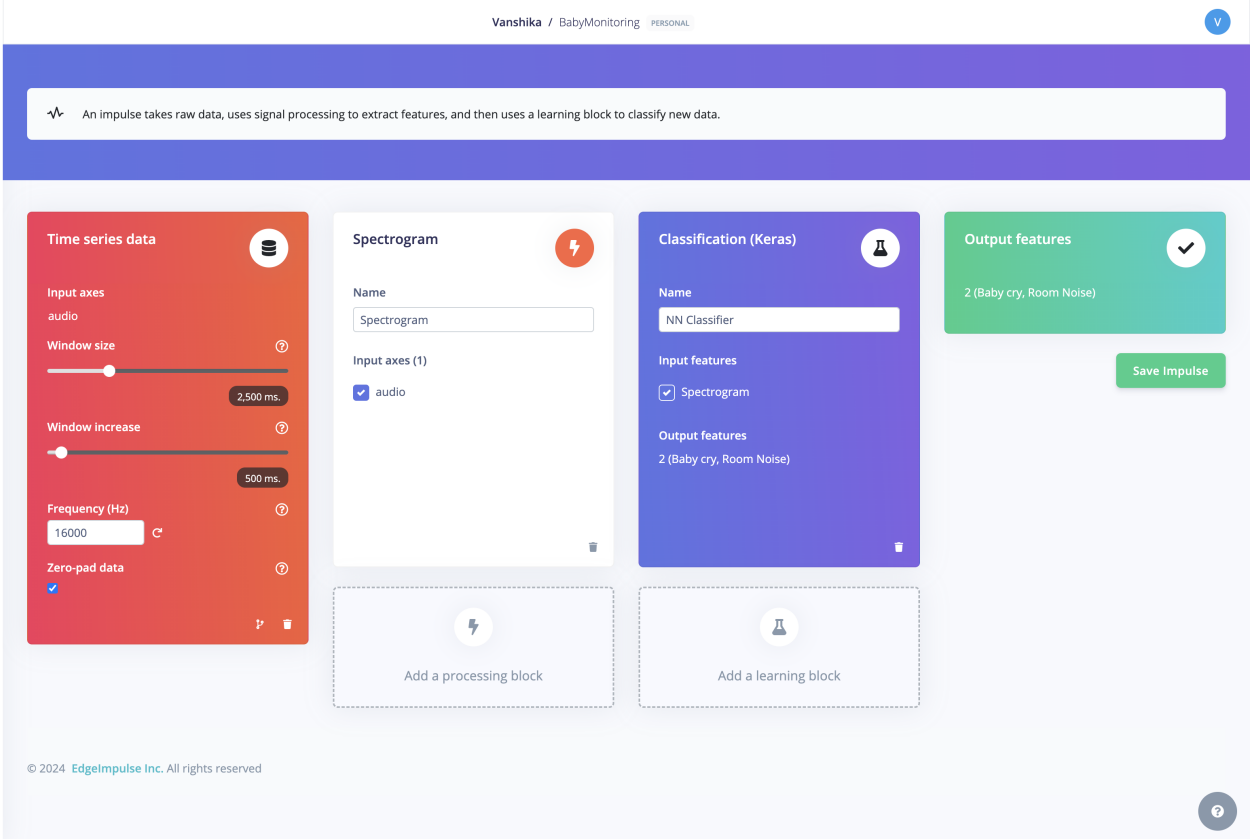
Edge Impulse is now linked to the UNO. We gathered a dataset containing both typical room noise and actual sounds of babies crying.

We used about 3 minutes 21 seconds worth of datasets.



Creating Impulse

The window sampling and window increase settings in our Edge Impulse account are 2500 ms and 500 ms, respectively, under the Create Impulse section. A spectrophotogram was set up as a preprocessing block.

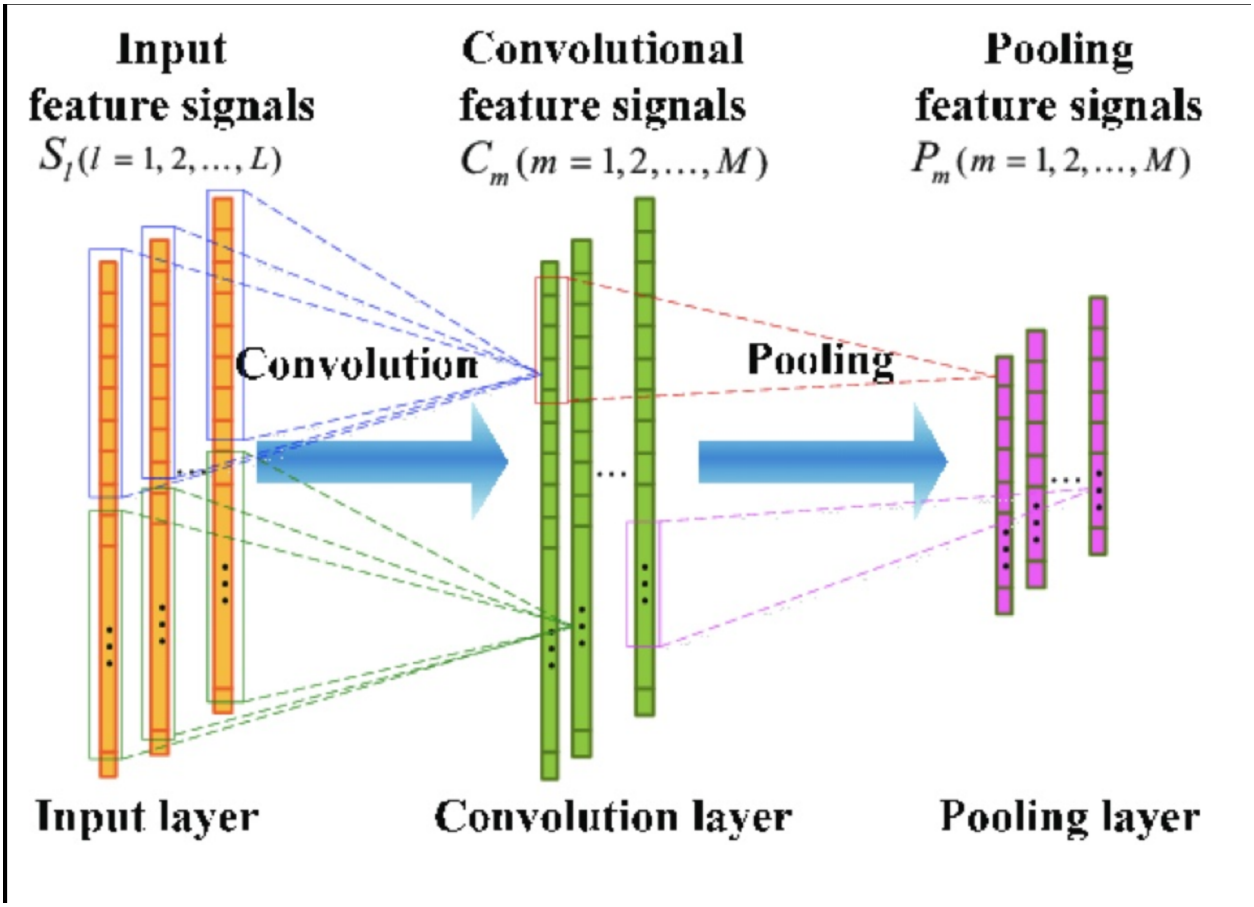


Neural Network Training

A convolutional neural network layer has been employed. A reshape layer is used to transform the input data from one dimensions into three dimension. Next, a Max Pooling 1 Dimensional convolution layer is employed.

1D Convolution Layer

The max pooling filtering and 1D convolution layer are shown in the diagram below.

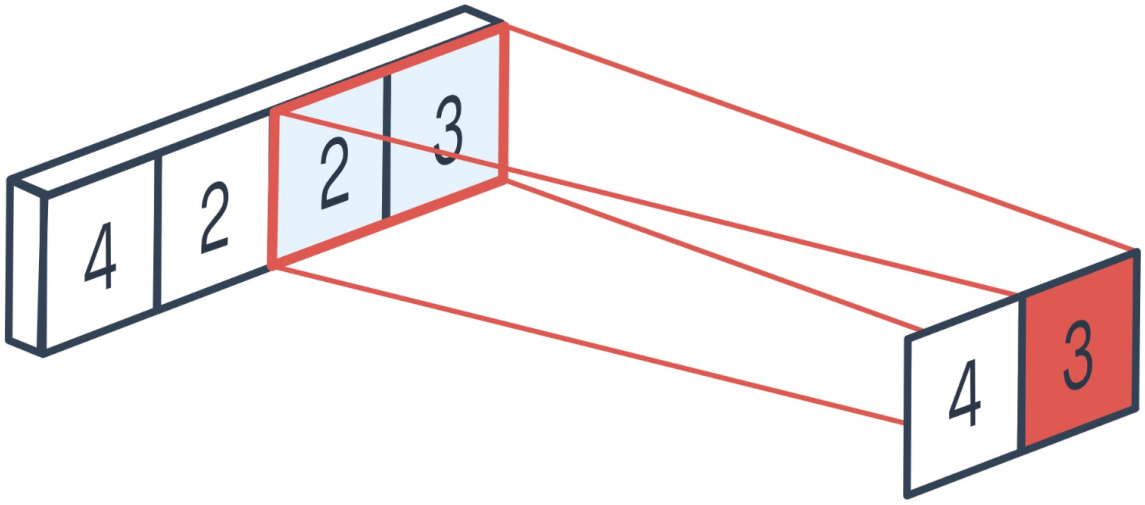


Max Pooling

In order to calculate the maximum in each unique window, the 1D max pooling block moves a pool (window) of a defined size over the incoming data with a set stride. The max pooling technique in 1D input data is shown in the diagram below.

The max pooling is configured as `MaxPooling1D(pool_size=2, strides=2, padding='same')`.

This indicates that there is a pool of size 2, and the greatest value in that pool is produced by using two indices. Additionally, it moves the pool layer twice in one direction due to the stride length of 2.



We attained a decent accuracy by configuring the training cycles to be 100 with a learning rate of 0.005.

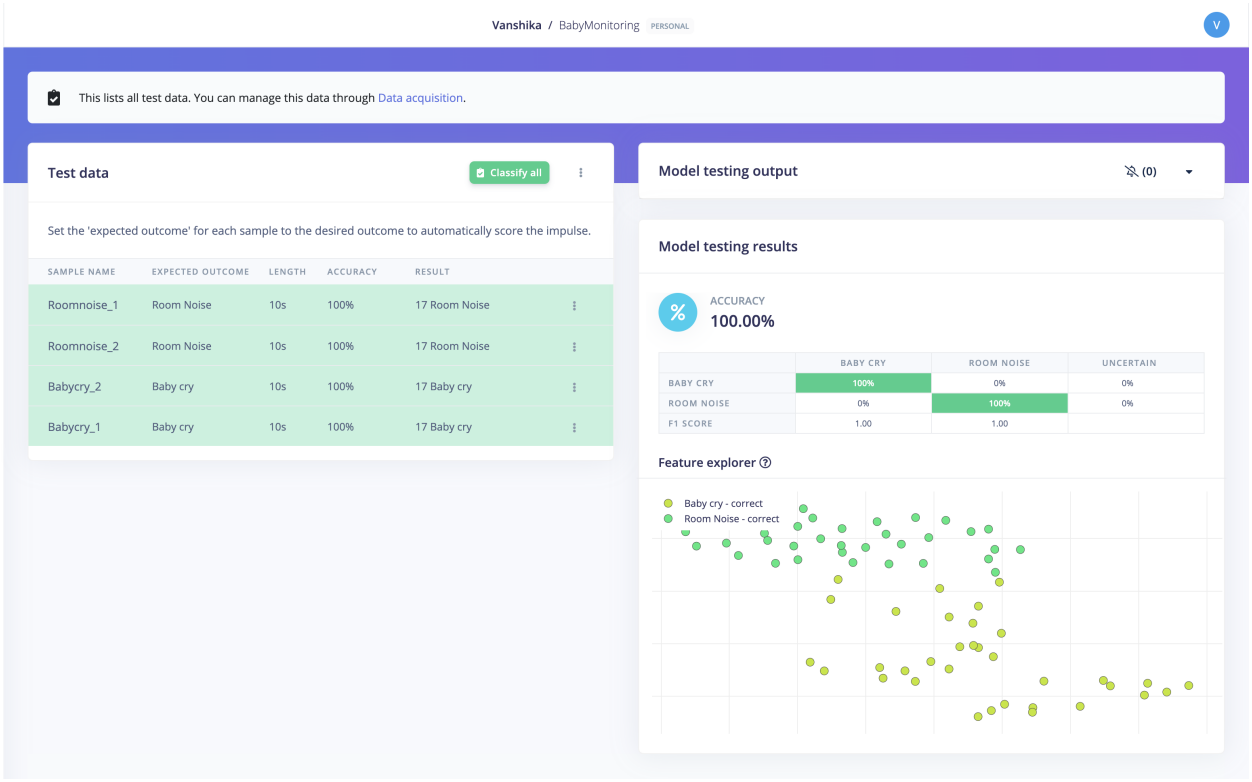
Results

Upon completion of training, we achieved 98.2% accuracy.

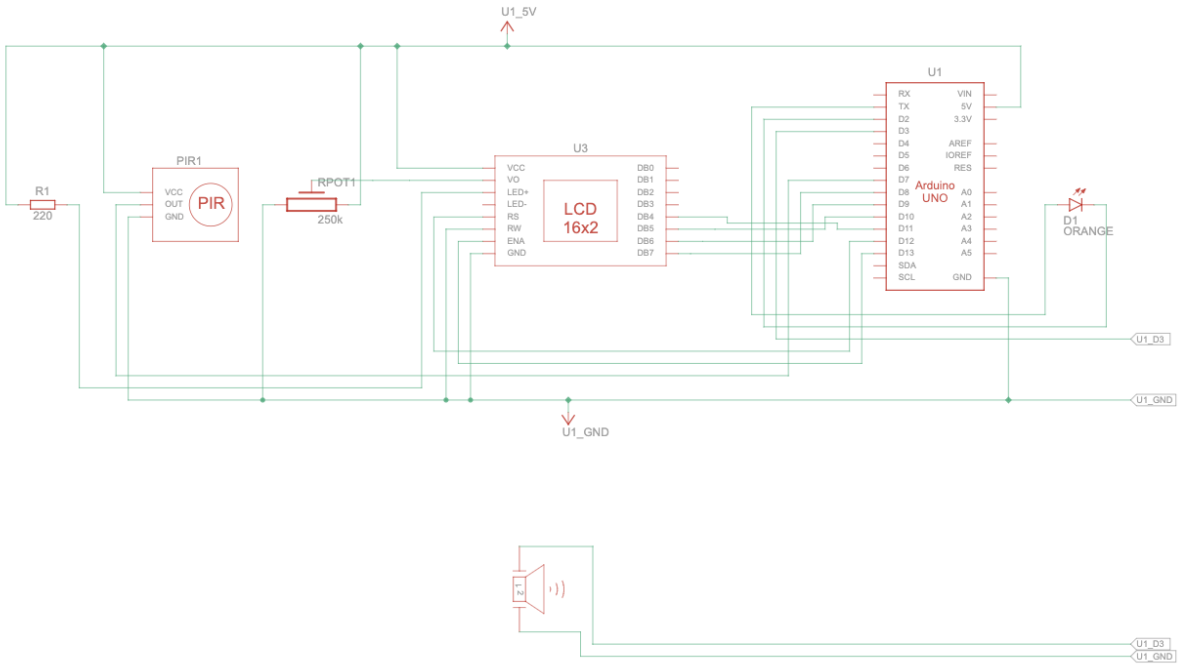
Model Testing

The model was tested on fresh data after it was trained with good accuracy. For testing, We used two datasets—one for room noise and one for baby cries. During

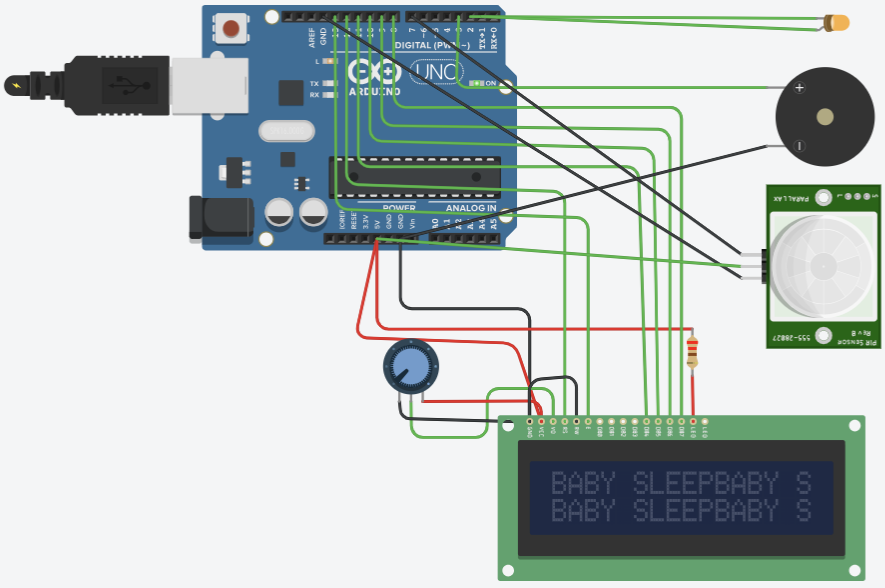
testing using hypothetical data, the model's accuracy was 100%.
The hardware deployment process comes next.

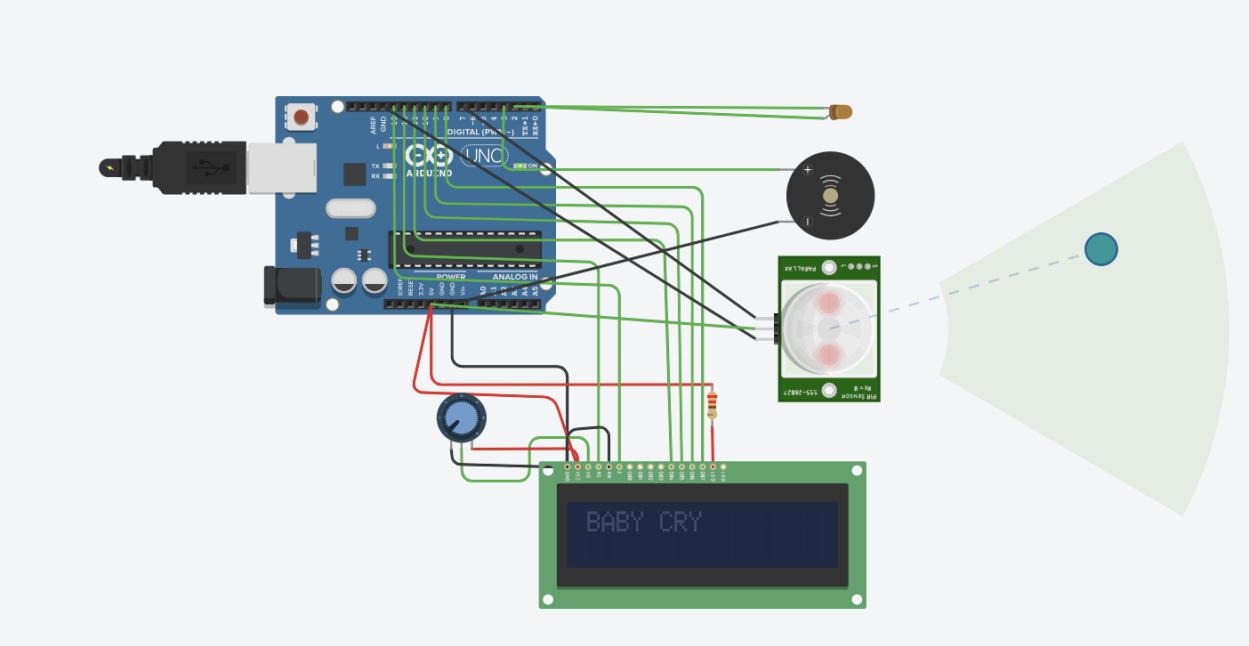


Circuit Diagram


U1_5V
U1
PIR1
R1
220
PIR
RPOT1
250k
U3
LCD
16x2
Arduino
UNO
D1
ORANGE
U1_D3
U1_GND
U1_GND
U1_D3
U1_GND

Simulation


BABY SLEEPBABY S
BABY SLEEPBABY S

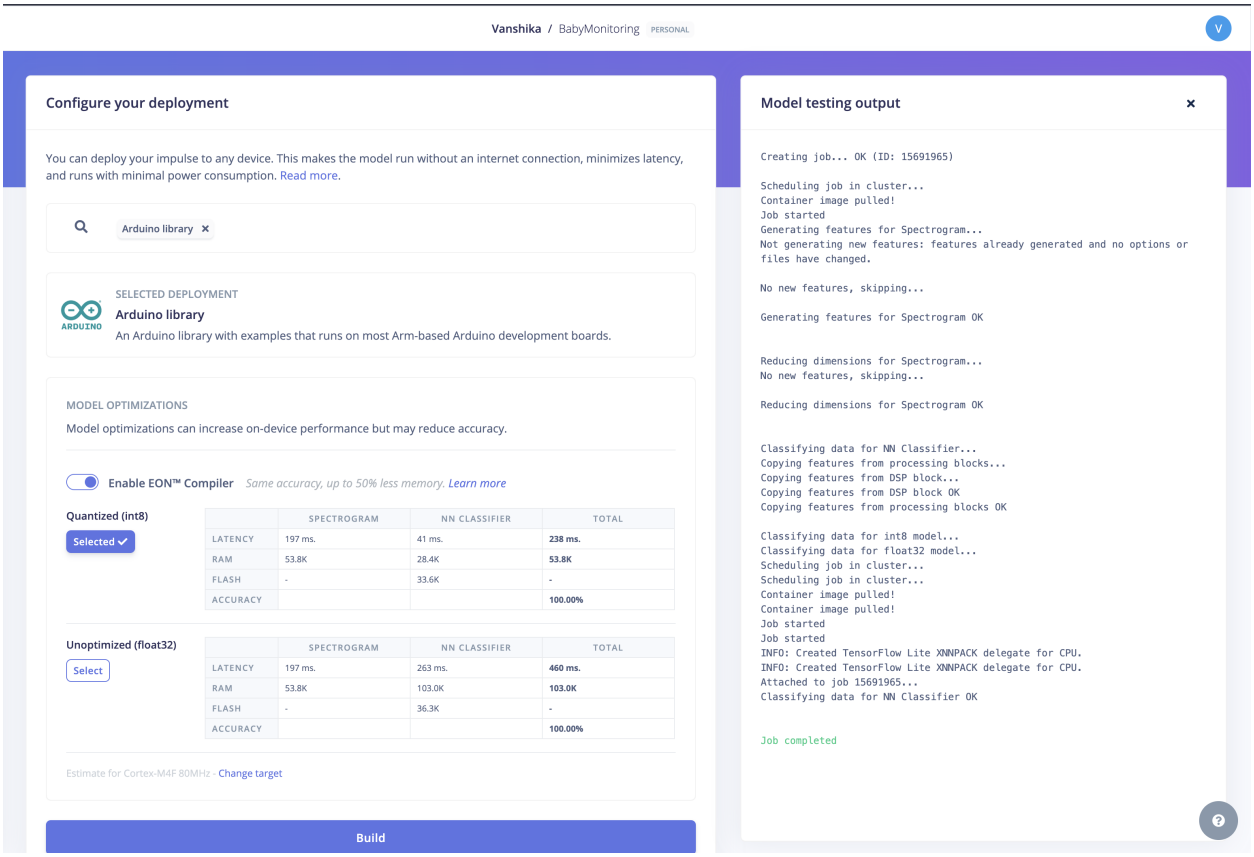


Deployment

Upon concluding the testing phase, proceed to the "Deployment" option and choose *Build firmware → Arduino Uno* to generate a downloadable firmware for flashing onto the board. The selected configuration is Quantized (Int8). In Edge Impulse, an additional option exists, allowing the utilization of the EON compiler to reduce resources, enhance accuracy, and achieve lower latency.

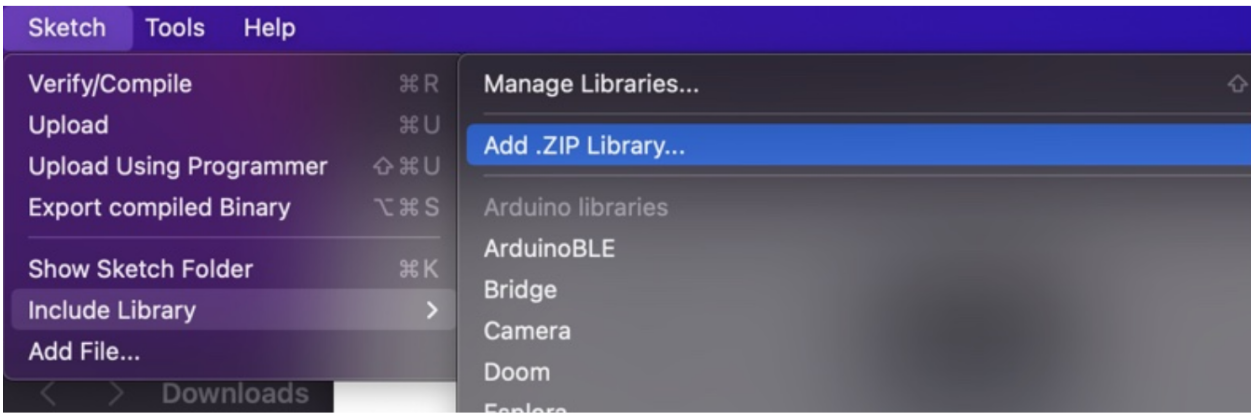
Upon the completion of the build process, the firmware is compressed into a Zip file. Initiate the "Flash mode" on the Uno by pressing the Reset button twice. Subsequently, we executed the .bat file (for Windows users) or run the Mac version (for macOS users) to flash the firmware. After successful flashing, open a new terminal window and execute the provided command to initiate inference on the device:

```
edge-impulse-run-impulse
```



The preceding step serves to verify the seamless operation of the model on real hardware. The main aim was to integrate the Portenta H7 into the baby swing rocker! To accomplish this, it is imperative to deploy the model as source code and incorporate our application atop it.

Navigating to the *Create Library* section in the Studio, opting for *Arduino*, and subsequently download the source code. Open the Arduino IDE and proceed to *Sketch → Include library* and select *Add .Zip Library*. Choose the downloaded Zip file on your machine.



Upon successful inclusion of the Library, navigate to *Examples* and choose `portent_h7_microphone_continuous`. We have authored the application code, building upon the default code provided in the example.

```
#ifndef EI_TFLITE_MODEL_OPS_DEFINES_H
#define EI_TFLITE_MODEL_OPS_DEFINES_H

#define EI_TFLITE_DISABLE_SOFTMAX_IN_U8     1
#define EI_TFLITE_DISABLE_SOFTMAX_IN_I16    1
#define EI_TFLITE_DISABLE_SOFTMAX_IN_F32    1
#define EI_TFLITE_DISABLE_SOFTMAX_IN_BOOL   1
#define EI_TFLITE_DISABLE_SOFTMAX_OUT_U8    1
#define EI_TFLITE_DISABLE_SOFTMAX_OUT_I16   1
#define EI_TFLITE_DISABLE_SOFTMAX_OUT_F32   1
#define EI_TFLITE_DISABLE_SOFTMAX_OUT_BOOL  1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_IN_U8     1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_IN_I16    1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_IN_F32    1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_IN_BOOL   1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_OUT_U8    1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_OUT_I16   1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_OUT_F32   1
#define EI_TFLITE_DISABLE_FULLY_CONNECTED_OUT_BOOL  1
#define EI_TFLITE_DISABLE_CONV_2D_IN_U8     1
#define EI_TFLITE_DISABLE_CONV_2D_IN_I16    1
#define EI_TFLITE_DISABLE_CONV_2D_IN_F32    1
```

```
#define EI_TFLITE_DISABLE_CONV_2D_IN_BOOL   1
#define EI_TFLITE_DISABLE_CONV_2D_OUT_U8    1
#define EI_TFLITE_DISABLE_CONV_2D_OUT_I16   1
#define EI_TFLITE_DISABLE_CONV_2D_OUT_F32   1
#define EI_TFLITE_DISABLE_CONV_2D_OUT_BOOL  1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_IN_U8     1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_IN_I16    1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_IN_F32    1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_IN_BOOL   1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_OUT_U8    1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_OUT_I16   1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_OUT_F32   1
#define EI_TFLITE_DISABLE_MAX_POOL_2D_OUT_BOOL  1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_IN_U8     1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_IN_I8     1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_IN_I16    1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_IN_F32    1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_IN_BOOL   1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_OUT_U8    1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_OUT_I8    1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_OUT_I16   1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_OUT_F32   1
#define EI_TFLITE_DISABLE_DEPTHWISE_CONV_2D_OUT_BOOL  1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_IN_U8     1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_IN_I8     1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_IN_I16    1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_IN_F32    1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_IN_BOOL   1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_OUT_U8    1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_OUT_I8    1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_OUT_I16   1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_OUT_F32   1
#define EI_TFLITE_DISABLE_AVERAGE_POOL_2D_OUT_BOOL  1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_IN_U8     1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_IN_I8     1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_IN_I16    1
```

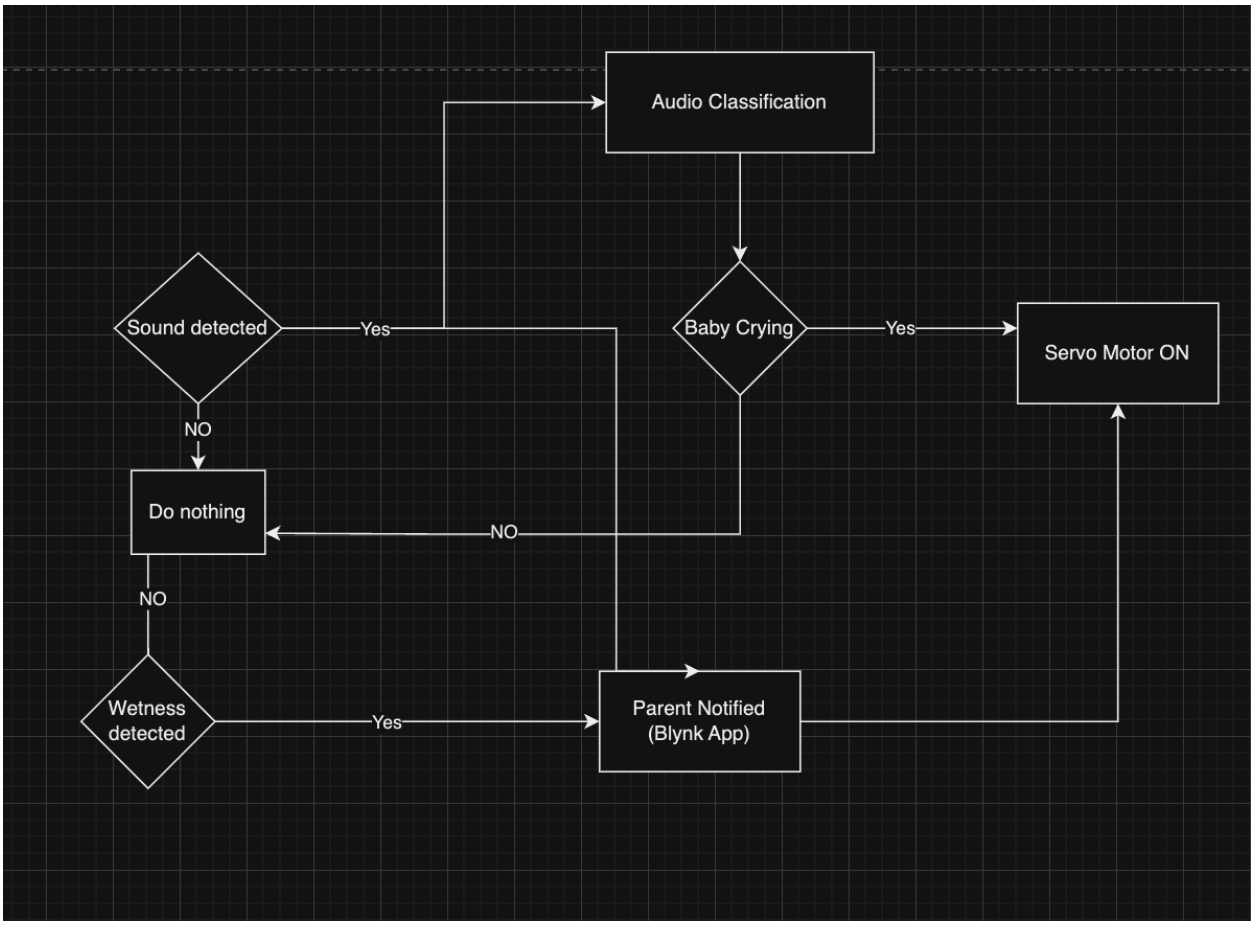
```
#define EI_TFLITE_DISABLE_STRIDED_SLICE_IN_F32    1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_IN_BOOL   1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_OUT_U8    1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_OUT_I8    1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_OUT_I16   1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_OUT_F32   1
#define EI_TFLITE_DISABLE_STRIDED_SLICE_OUT_BOOL  1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_IN_U8     1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_IN_I8     1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_IN_I16    1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_IN_F32    1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_IN_BOOL   1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_OUT_U8    1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_OUT_I8    1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_OUT_I16   1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_OUT_F32   1
#define EI_TFLITE_DISABLE_TreeEnsembleClassifier_OUT_BOOL  1

#endif // EI_TFLITE_MODEL_OPS_DEFINES_H
```

Application Layer Integration

The application code is designed with logic to trigger a relay connected to the baby swing rocker's motor. The provided flowchart illustrates the sequence of operations. Specifically, the application code is programmed to activate the baby swing rocker for a duration of 20 seconds each time it detects the sound indicative of a crying baby.

Hardware Integration

The Arduino Uno is linked to the 5V DC Relay module. The Common pin on the relay is linked to the battery's ground, and the NO pin on the relay is connected to the ground of the motor in the baby swing rocker. Meanwhile, the motor's Vcc is directly connected to the positive terminal of the battery.



Summary

A pioneering solution has been introduced employing TinyML technology to address the limitation mentioned earlier. An Arduino, equipped with audio inferencing capabilities, discerns the sound of a crying infant amidst background noise and autonomously activates the baby swing rocker motor.

Utilizing Edge Impulse account settings, with window sampling and window increase set at 2500 ms and 500 ms respectively, a spectrophotogram is incorporated as a preprocessing block in the neural network training. The sequential neural network comprises a reshape layer that transforms input data from two dimensions to one, followed by the application of a 1D Max Pooling convolution layer.

The model undergoes training using the testing dataset, leading to the generation of stimulations and subsequent successful deployment.