Human Emotion Detector
Configuration steps to setup the Raspberry pi device

Hardware devices used:

1. Raspberry pi 3 Model 3B with built in Wifi and Bluetooth connectivity.
2. Micro SD Card - With New out of box Raspi software
3. Grove base hat for Raspberry pi
4. Galvanic Skin response sensor (3.3V/5V) from Grove Seeed
5. Ear-clip heart rate sensor (3V/5.25V) from Grove Seeed

Initial setup:

Step 1: Insert the Micro SD card that is pre-loaded with NOOBS (New out of the box software) into the micro SD card slot located on the bottom of the Raspberry device

Step 2: Connect the 2.5A power adapter to the board. When the device is powered, the Raspberry pi will boot and a menu to install the the operating system will be displayed

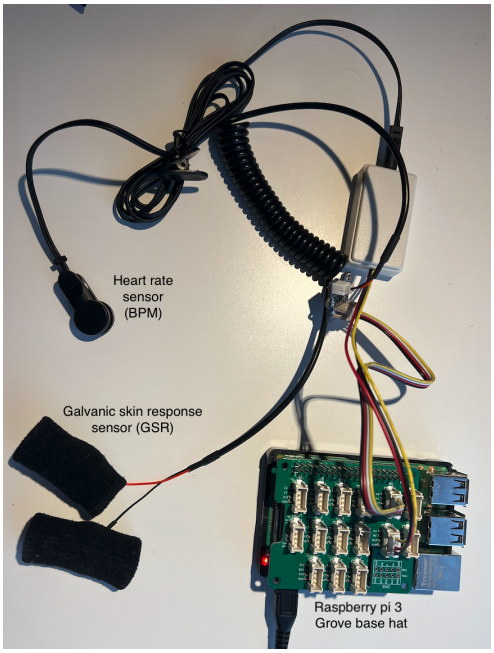
Step 3: Install the Raspbian OS and the device will reboot automatically after installation

Step 4: Set up the wifi network and password and using the IP address, login to the device using SSH.

Step 5: Attach the Grove base hat onto the GPIO pins of the Raspberry pi board

Step 6: The software configuration of raspberry pi is done using the "**sudo raspi-config**" command. Configurations like setting raspberry pi password, allowing access to VNC, SSH, GPIO, Serial communication, I2C etc are specified here.

Step 7: Connect the two sensors to the analog ports A0 and A4 of the grove base hat as shown in Figure 1. The data from this port can be read in python using the ADC converter of the grove python library [resistance = adc.read(0)]



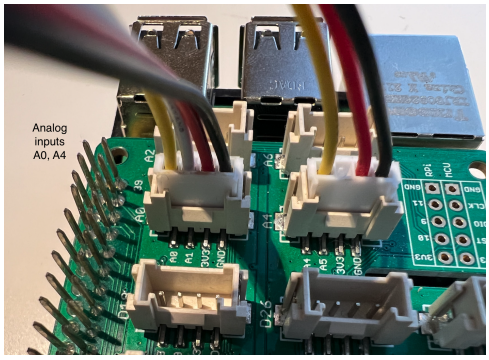


Fig 1: Hardware components and connection

Installation of sensor python library:

Install the Python library for Seeedstudio Grove Devices on embedded Linux platforms like Raspberry pi using the following commands

```
git clone https://github.com/Seeed-Studio/grove.py
cd grove.py
sudo pip3 install .
```

Important Note:

Because ST32 series chips are out of stock globally, they were switched to the MM32 chip. This change affects the way the data from the ports are being read. It creates an error saying "Check if I2C enabled". The workaround for this issue is as follows:

1. Change the permissions on adc.py located in /usr/local/lib/python3.7/dist-packages/grove/adc.py by typing sudo chmod 666 /usr/local/lib/python3.7/dist-packages/grove/adc.py
2. In the adc.py file, go to approximately line 57, and change the 0x04 to 0x08 as shown below
   from: def __init__(self, address = 0x04):
   to: def __init__(self, address = 0x08):
3. Save the adc.py file.
4. Change the permissions back on adc.py by typing sudo chmod 644 /usr/local/lib/python3.7/dist-packages/grove/adc.py.

AWS IoT client:

Step 1: In the AWS Management console, navigate to AWS IoT core and create a Thing

Step 2: When a thing named "Raspberry_pi" is created, 'Auto-generate a new certificate' is chosen to generate the certificates. Download these certificates and save them in a secure location of the Raspberry pi device.

Step 3: The paths of these certificates are provided in the deviceCode/Training/datacollection.py script that establishes connection between the local device and AWS cloud.

Step 4: Move the datacollection.py file to the path of the cloned repository (grove) for easier access of their libraries.

Step 5: Run the script using "**python3 datacollection.py**" to start collecting sensor data from the user based on the video stimuli.

User interface setup:
These steps are followed during the inference phase of the project

Step 1: In the terminal, run "**python3 deviceCode/Inference/Initiation.py**" to run the flask application. A port number is assigned in the localhost.
Step 2: Install a VNC Server in the raspberry pi and VNC Viewer in the laptop to access the GUI of the device.
Step 3: Open a browser, and type the specified local host address. Click on "Detect your emotion" to start the inference process. The detected emotion is shown in the same window as shown in Figure 2.

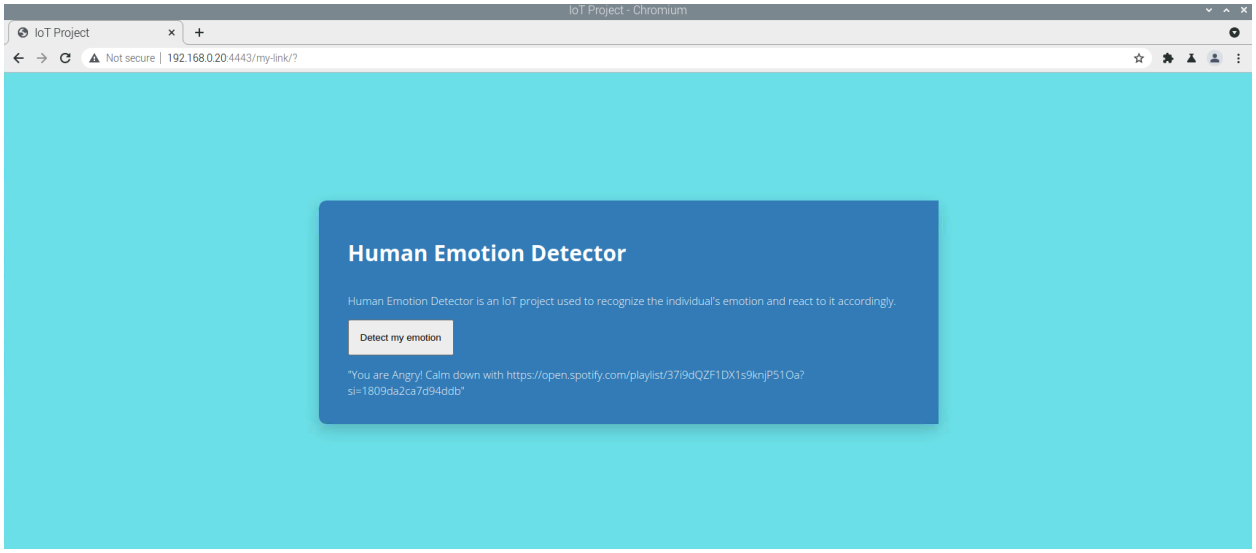
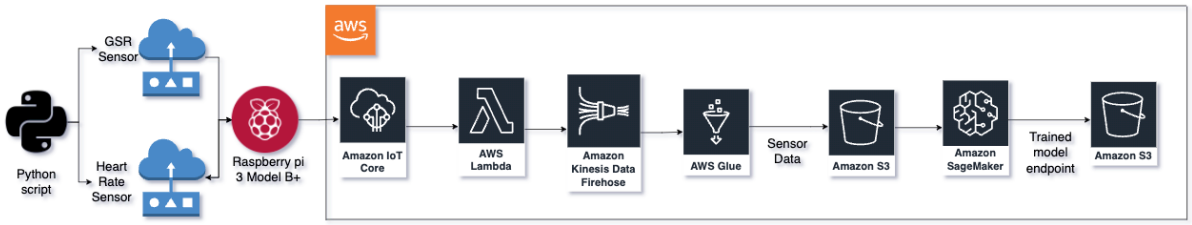


Fig 2: User Interface

Configuration steps to run the cloud services

Note:
Managing the access to AWS Resources is performed in the IAM Management console of AWS Cloud

Training:



AWS IoT Core:

1. Under Secure → Policies of AWS IoT core, create a policy which gives access to all topics with the following prefix "iotsensors/*" to publish, subscribe, receive, or connect. Figure 1 shows the sample policy used for the training phase.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:[redacted]:topic/iotsensors/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:[redacted]:topicfilter/iotsensors/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:[redacted]:client/iot-sensor"
      ]
    }
  ]
}
```

Fig 1: Policy

2. Create a Thing named “Raspberry pi” as a single device and auto generate certificates. When creating a thing, select the policy created in step 1. Store these downloaded certificates securely in the raspberry pi device.
3. Under Act → Rules, create a rule with the rule query statement [SELECT * FROM 'iotsensors/train'] and setup the action of triggering a Lambda function. The configuration of rules is shown in Figure 2.

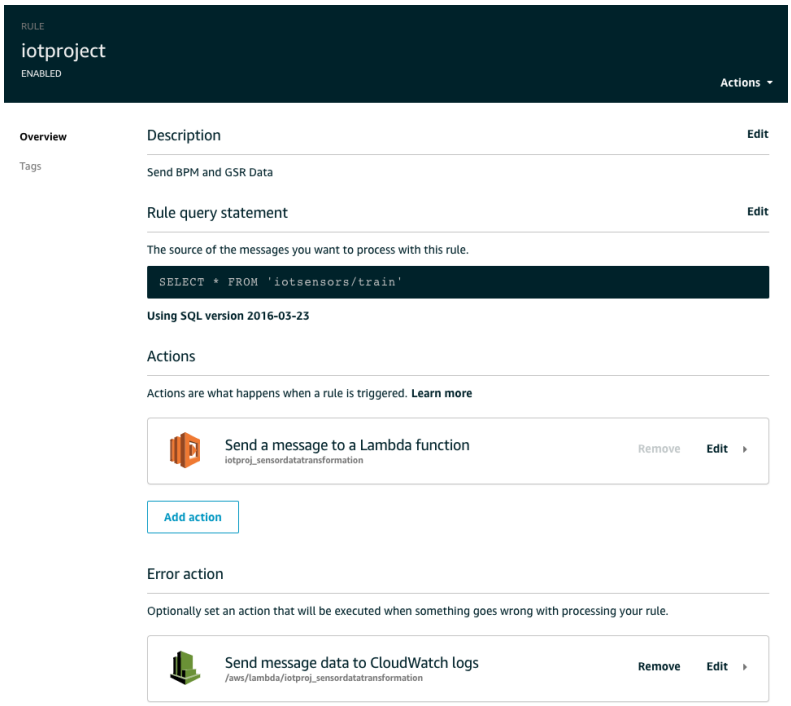


Fig 2: Rule setup in AWS IoT Core

AWS Lambda:

1. Create a serverless lambda function named “iotproj_sensordatatransformation” that uses python3.9 runtime.
2. Use the content of code from **cloudCode/Training/lambda_function.py** and deploy it.
3. The Lambda function helps to mathematically round the raw sensor data with up to 3 points of precision.

Amazon Kinesis Data Firehose and AWS Glue:

1. Create a delivery stream that captures the sensor data data from AWS Lambda and stores them in a S3 bucket “iotproj-sensordata”. The data from Lambda function is in a Json file format. Inorder to convert to a parquet file format, use the “Convert record format” option and choose the output format as “Apache Parquet”. The configuration is shown in Figure 3.

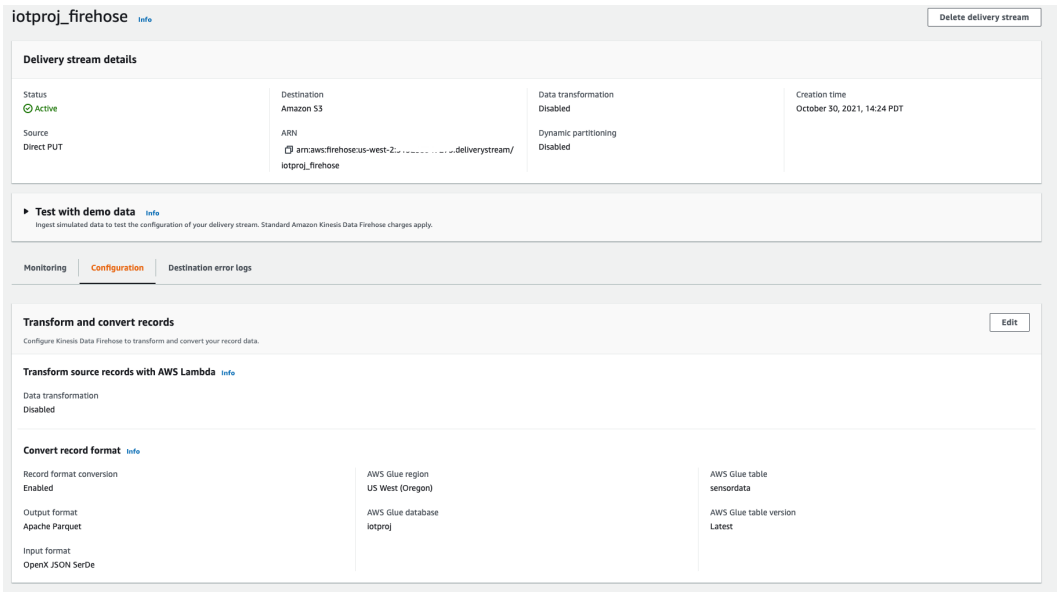


Fig 3. Firehose delivery stream

2. In AWS Glue, create a table using a sample sensor data json file. Run a crawler on top of it to fetch the data fields as shown in Figure 4. Add this table information to the AWS Firehose delivery stream on creation.

Schema

	Column name	Data type
1	gsr	double
2	bpm	double
3	mood	string
4	date	string
5	time	string

Fig 4. AWS Glue schema

Amazon Sagemaker:

1. Create a sagemaker notebook CPU instance as shown in Figure 5.

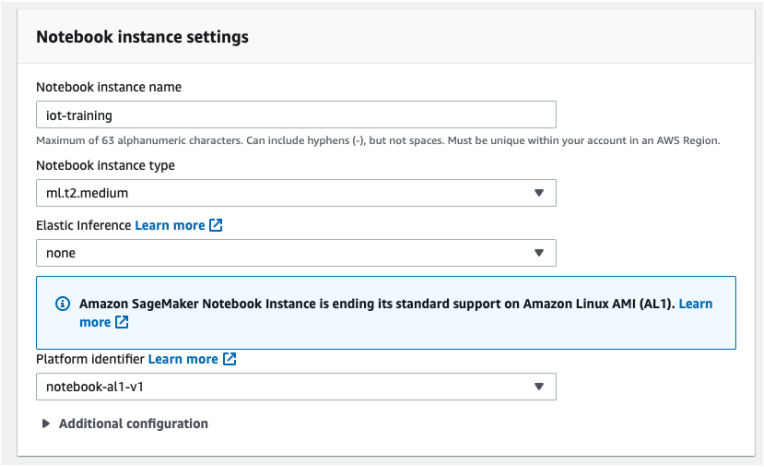


Fig 5. Creating a Sagemaker notebook instance

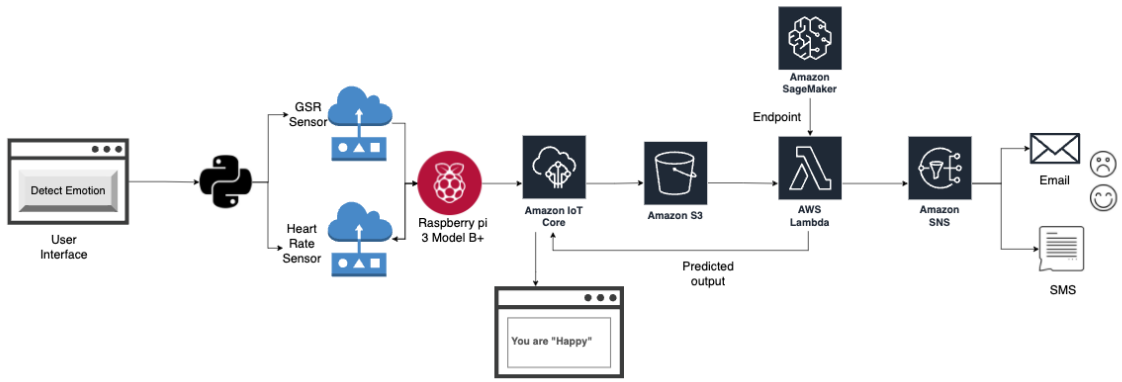
2. Additional information can be provided to automatically stop the jupyter notebook when it is idle after a certain preset time. This script is configured under the Notebooks → Lifecycle configurations of Amazon Sagemaker service. '**Sagemaker_autostop.sh**' bash script provided under cloudCode/Training can be used to stop the jupyter notebook after 2000 seconds of no activity.
3. Use the jupyter notebook instance **cloudCode/Training/datacollection.ipynb** to collect the sensor data stored in s3 bucket. The script is set to perform data collection, aggregation, visualization.
4. The notebook **cloudCode/Training/training_sagemaker.ipynb** is used to train the aggregated data using a linear learner algorithm. Appropriate S3 bucket names must be provided to save the trained model and its corresponding log files.
5. Once a model is trained, a sagemaker endpoint is created under Inference → Endpoints as shown in Figure 6. The name of this endpoint will be used in the inference stage of the project.

Fig 6. Sagemaker endpoint

Python libraries used in Amazon Sagemaker:

- **Pyarrow -** Used to read/write parquet files with pandas.
- **Seaborn -** A Python data visualization library based on matplotlib that is used to visualize the distribution of sensor data for varied BPM and Sweat rate.
- **Sklearn -** A machine learning library in Python that is fit for statistical analysis and modeling
- **Sagemaker -** An open source library for training and deploying machine-learned models like Linear learner on Amazon SageMaker.

Inference:



User Interface:

1. A website that is built using Flask, Html and CSS is an interactive user interface to start the inference process.
2. On clicking the “Detect my emotion” button found in the website, deviceCode/Inference/initiation.py is triggered, which inturn runs the deviceCode/Inference/inference.py.
3. The user is prompted to wear the two biomedical sensors on the fingertips. After collecting the data for 5-10 seconds, the data is sent onto the cloud using MQTT protocol for further processing.

AWS IoT Core:

1. The biomedical data sent from the GSR and Heart rate sensor are published to a new topic “iotsensors/infer/result” sent from the **deviceCode/Inference/inference.py.** AWS IoT core is configured to trigger a new action on receiving messages from this particular topic as shown in Figure 7.

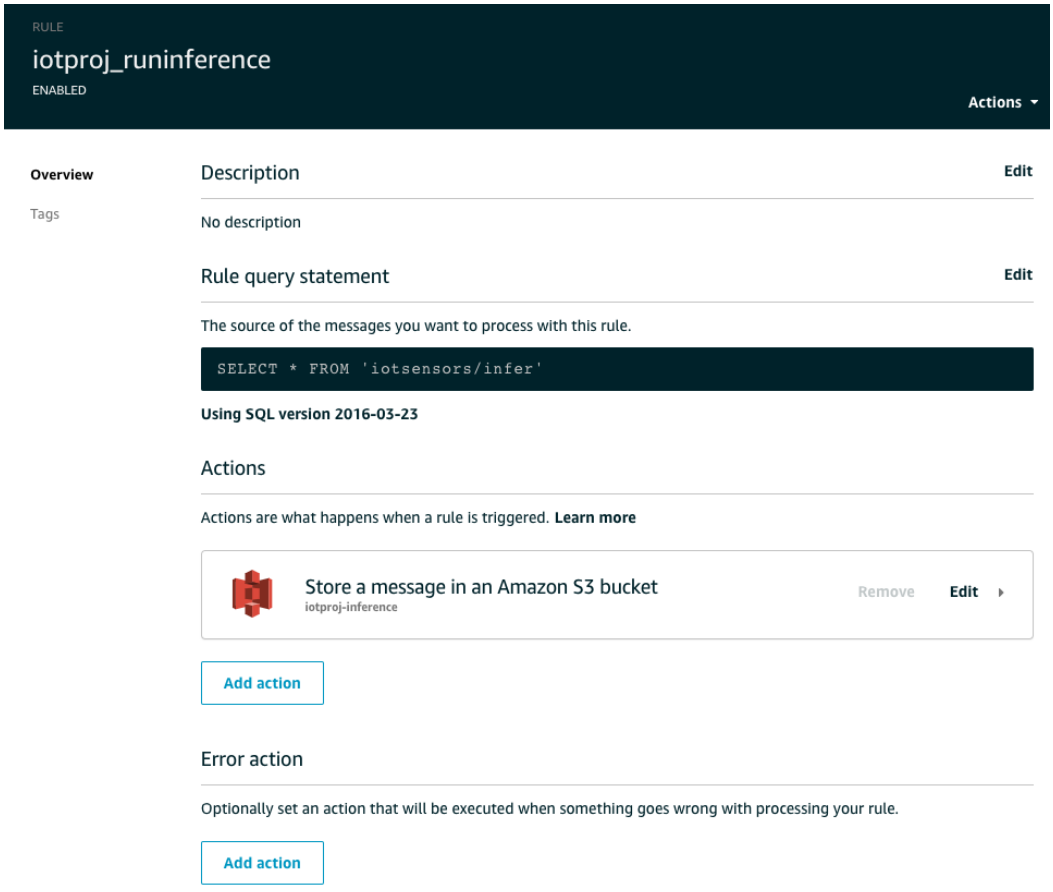


Fig 7. Rules for Inference phase

2. Actions are configured to store all the incoming messages in an Amazon S3 bucket named “iotproj-inference”.

AWS Lambda

The following steps occur sequentially in the lambda function - cloudCode/Inference/lambda_function.py

1. When a new file is uploaded in s3 bucket ‘iotproj-inference’, lambda function is triggered. This can be setup as shown in Figure 8.

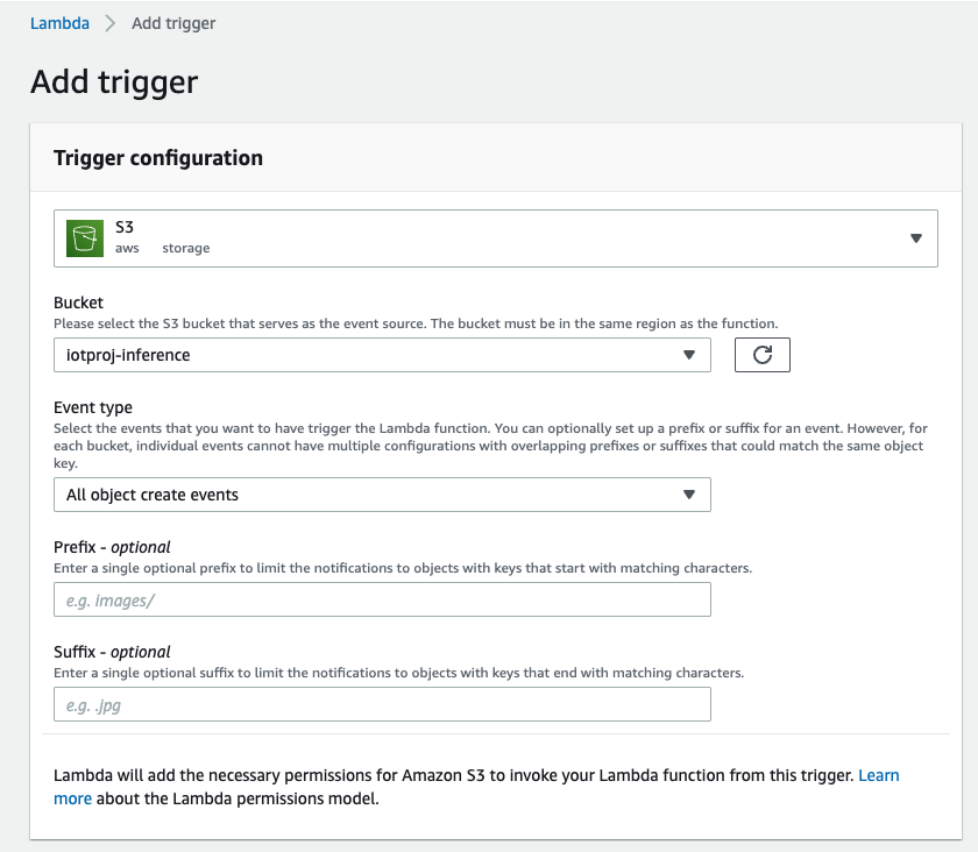
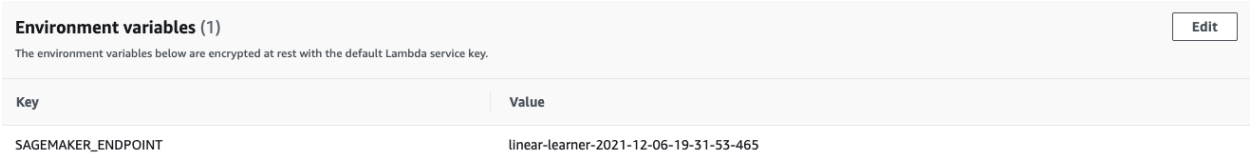


Fig 8: Lambda function trigger

2. Read the contents of uploaded sensor data file using the get_object function
3. Invoke sagemaker endpoint. To specify the specific endpoint to use, environment variables are set up as shown in Figure 9.



4. The invoke_endpoint function returns the output of the model ie. prediction emotion in numerical format.
5. Retrieve the categorical value from numerical output when they are mapped as follows,

```
emotion = { 0: "Angry", 1: "Happy",  2: "Sad"}
```

6. Using this result, fetch the appropriate recommendation message from another json file stored in the S3 bucket ‘iotproj-inference/output.json’. The contents of the json file is shown as follows,

{ "Happy":"You are Happy! Stay Happy and listen to https://open.spotify.com/playlist/37i9dQZF1DXdPec7aLTmlC?si=f720fb2b9b2e4077",

"Sad":"You are Sad. I think you will like listening to https://open.spotify.com/playlist/4nch8ph3KYgdHY3boJeMJZ?si=9582633714294e99",

"Angry":"You are Angry! Calm down by listening to https://open.spotify.com/playlist/37i9dQZF1DX1s9knjP51Oa?si=1809da2ca7d94ddb"}

7. Publish the message to a new topic in AWS IoT core “iotsensors/infer/result”
8. Publish message to SNS topic ‘iotproj-emotion’

AWS SNS

Create a topic named “iotproj-emotion” that has subscriptions of phone number and an email address as shown in Figure 9.

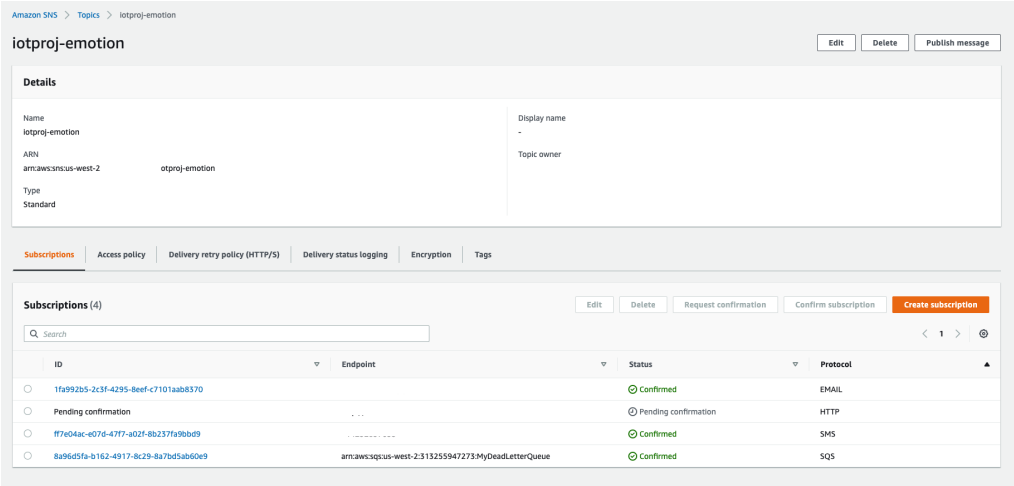


Fig 9. SNS Topic

AWS IoT Core

1. The message from the topic “iotsensors/infer/result” is published to the raspberry pi device.
2. The raspberry pi’s **deviceCode/Inference/Inference.py** subscribes to the result and waits for a predicted emotion.
3. On receiving the output, the detected emotion + music recommendation is shown on the website that is hosted locally.