Team 1 – Cloud Navigators – Project Design Report

GROUP MEMBERS:

- ADHEIK DOMINIC
- TANMEET BUTANI
- JASON PAUL

1. ***Introduction:***

   Smoking is a dangerous and addictive habit that can lead to a range of health problems, including lung cancer, heart disease, and stroke. Despite the well-known risks, many smokers find it difficult to quit, often trying and failing multiple times. Quitting smoking can be a challenging process that requires support, encouragement, and guidance. To address this issue, we propose the development of StubACT a smart-watch and thermal-camera based inference system that tracks when the wearer is smoking and notifies them on the smart-watch.

2. ***System Design and Implementation:***

   The design of the system is very specific to this goal and a watch has been chosen so that it cannot connect to a smartphone while being utilized for this purpose. There are certain factors that were considered when designing this system, they are as follows:

   - Size of the thermal-camera unit. To make sure it is not socially awkward to wear the device.
   - Weight of the thermal-camera unit. To ensure comfort in wearing the device while performing daily activities.
   - Power consumption of the whole system including the watch. To ensure the longevity of the system between consecutive charges.

   Considering these factors, the project system is divided into two units:

   a. The Human-Interaction unit – Smart-watch.
   b. The Inference unit – Thermal-camera system.

   The Human-Interaction unit (a) is chosen to be an Android smartwatch i.e., ***<Enter watch name here>***. This was chosen because 1. It is a cheap enough watch with Bluetooth Low Energy, 2. It is open source, has great community support and 3. It has enough resources besides BLE such as Heart Rate Monitor, Accelerometer which support the future scope of this project.

   The Inference unit (b) is chosen to be an ESP32 based IoT board with thermal-camera module, because 1. It is a cheap and a modular IoT platform to base the unit on, 2. It also supports Bluetooth Low Energy protocol essential to the Power factor of the System Design as well as all the other sensors on board are very low energy contributing to the Power factor as well, 3. It supports on

board storage to collect real-time thermal data from the system with reinforced inferencing via unit (a) to later have some data to train the model on.

The rest of the implementation is software-based. The embedded firmware running on the Inference Unit (b) will capture and infer thermal data using the thermal module attached on board. It will then post a notification via BLE to unit (a) to reinforce the data from the wearer whether they were actually smoking or not, make a note of this data, and along with the timestamp, thermal frame in 768 csv values and smoking inference 00000 – no, 11111 – yes, will store the received data onto the SD Card mounted to the device. This data will be used to train a model off the system.

3. ***Risks:***

Certain things are a first for all teammates. Like writing an app for WearOS, Tanmeet has certain experience working with ESP32 WROOM BLE and Wi-Fi. Adheik and Jason have experience with electronics. This may be a bottleneck, as it is a steep learning curve writing apps that use BLE on WearOS.

As far as project completion goes, we are certain of completing the project to a proof-of-concept stage. A complete production ready system can be targeted for, but we cannot think about it before getting to a proof-of-concept stage. Besides, the more important aspect is to collect the data, and use that data to train model that can infer smoking detection. Getting to this stage is what we are focusing right now.

4. ***Conclusion:***

All in all, the project is an excellent proof-of-concept of a tool that can be used for people who are trying to quit smoking. Also, the gestures captured in this model can also be used to train a model that detects food as in the thermal frame, and can also help people who are trying to control their eating habits.

5. ***References:***

ESP32 Board - https://www.amazon.com/HiLetgo-ESP-WROOM-32-Development-Microcontroller-Integrated/dp/B0718T232Z
ESP32 Thermal Camera Module – https://www.adafruit.com/product/4469
ESP32 RGB Camera Module - https://www.amazon.com/Arducam-Module-Megapixels-Arduino-Mega2560/dp/B012UXNDOY/ref=sr_1_2?crid=224GE0OTX3EI7&keywords=Arducam+Mini+Module+Camera+Shield+with+OV2640+2&qid=1683593750&s=industrial&sprefix=arducam+mini+module+camera+shield+with+ov2640+2%2Cindustrial%2C116&sr=1-2
ESP32 Battery - https://www.digikey.com/en/products/detail/sparkfun-electronics/PRT-13851/6605199?s=N4IgTCBcDaIAoCUAqBaAjAZgBwFY0gF0BfIA
BLE on WearOS - https://developer.android.com/guide/topics/connectivity/bluetooth/ble-overview
Paper for Reference from Habits Lab - https://habitslab.github.io/publications/smartact/

6. ***Documentation***:

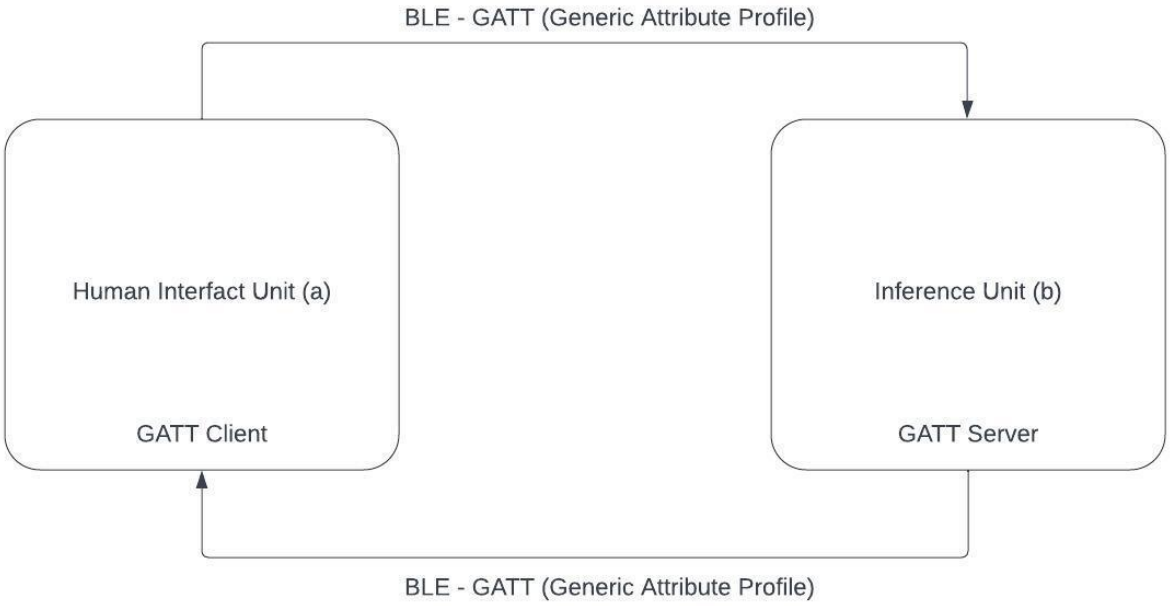

Image 1 – Top Level System Diagram

```
class BluetoothLeService : Service() {

    private val binder = LocalBinder()

    override fun onBind(intent: Intent): IBinder? {
        return binder
    }

    inner class LocalBinder : Binder() {
        fun getService() : BluetoothLeService {
            return this@BluetoothLeService
        }
    }
}
```

Image 2 – Code Snippet to Invoke GATT Service on WearOS

```
/*
    Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/master/cpp_utils/tests/BLE%20Tests/SampleServer.cpp
    Ported to Arduino ESP32 by Evandro Copercini
    updates by chegewara
*/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

// See the following for generating UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE work!");

  BLEDevice::init("Long name works now");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ |
                                         BLECharacteristic::PROPERTY_WRITE
                                       );

  pCharacteristic->setValue("Hello World says Neil");
  pService->start();
  // BLEAdvertising *pAdvertising = pServer->getAdvertising();  // this still is working for backward compatibility
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);  // functions that help with iPhone connections issue
  pAdvertising->setMinPreferred(0x12);
  BLEDevice::startAdvertising();
  Serial.println("Characteristic defined! Now you can read it in your phone!");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(2000);
}
```

Image 3 – Code Snippet for BLE Server on ESP32

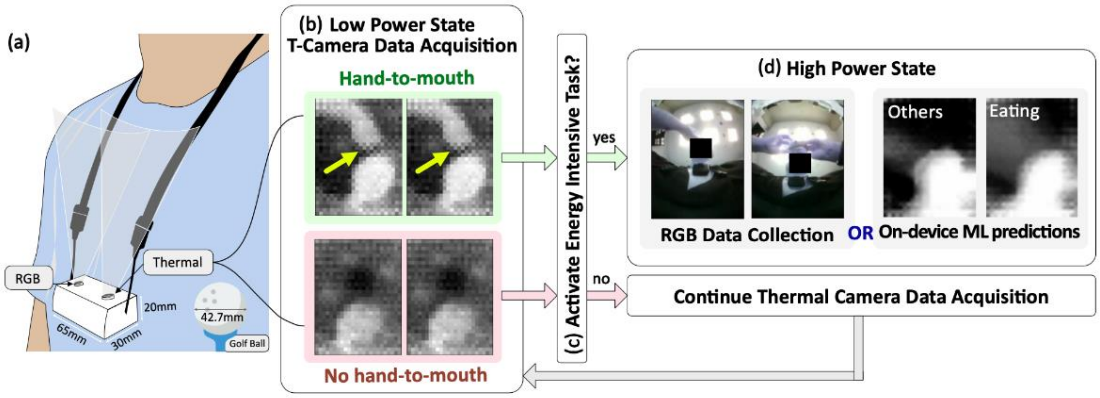

Image 4 – Thermal Camera Hand-To-Mouth Gesture Detection.

Image 5 – ESP32-WROOM-32 Board

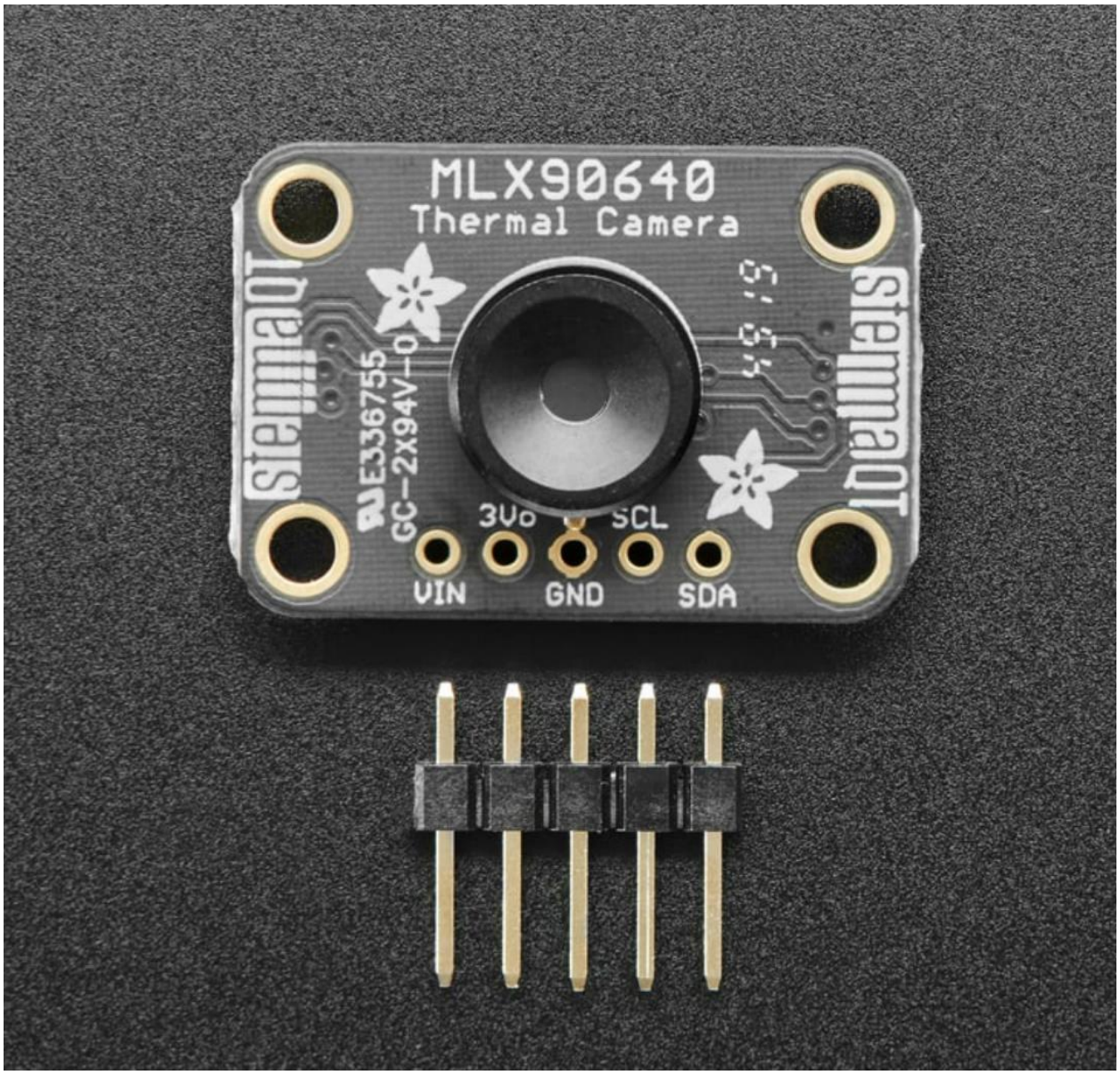

Image 6 – Thermal Camera Breakout Module.

Image 7 – RGB Camera

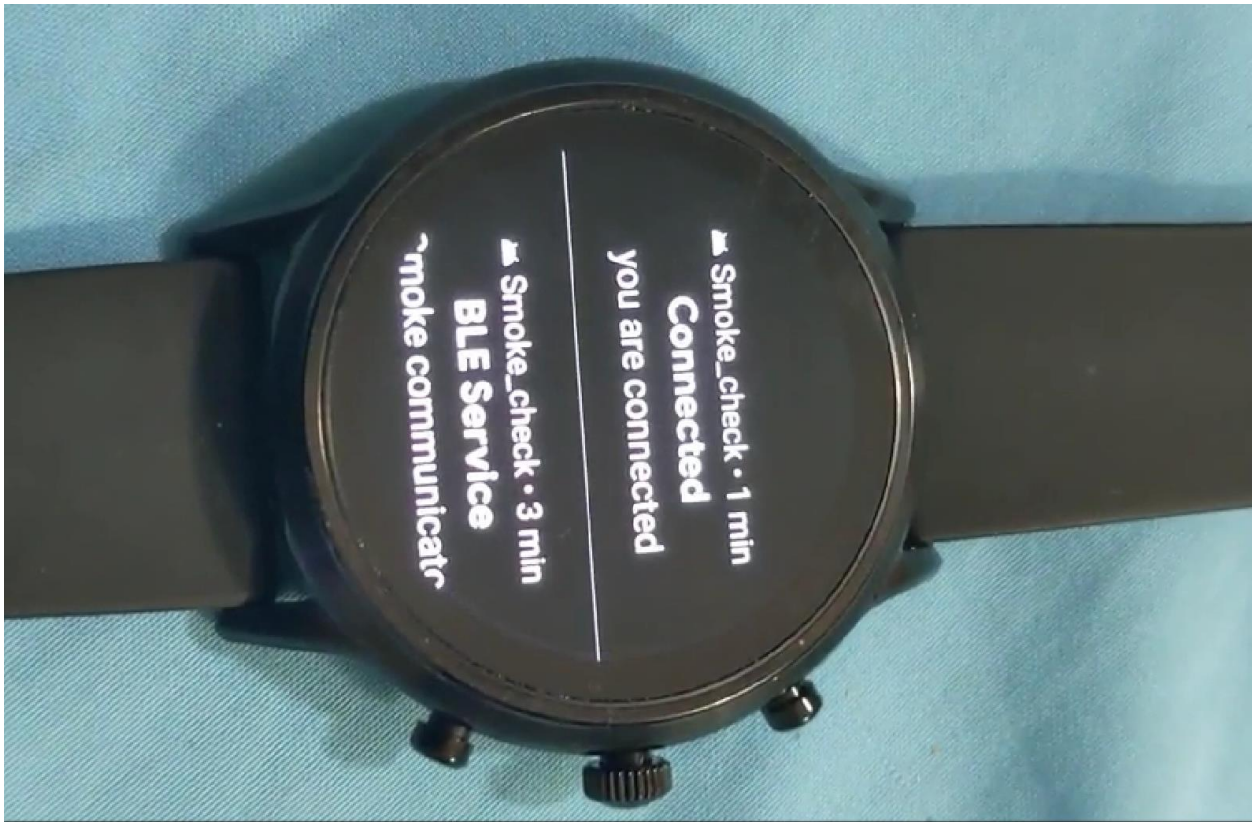

Image 8 – Proof of Concept app sample using BLE

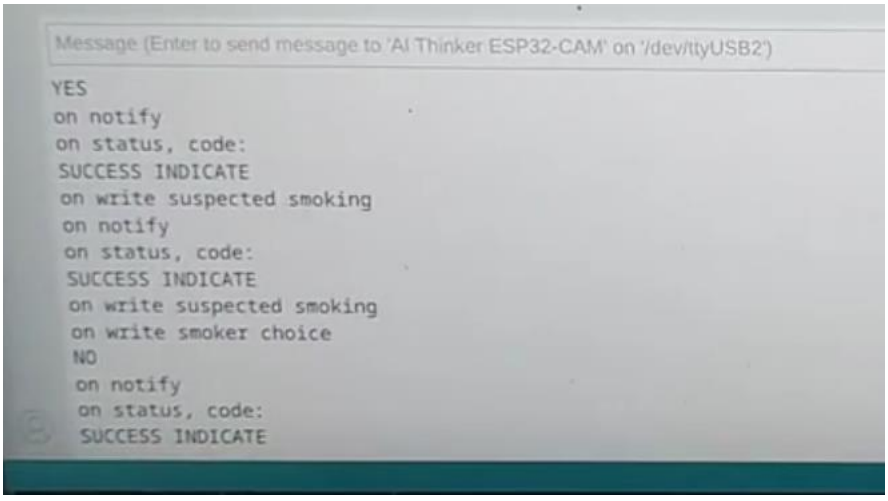

Image 9 – Serial Monitor on running the sample server