EEE3999 Technical Answers for Real World Problems (TARP)

Team Name	The Invincibles
Project Title	Sign Language translator
Team Members (Reg.No & Name)	1.Muskan Agarwal 16BEE0143 2.Aashi Kapoor 16BEE0130 3.Ananyo Banerjee 16BEE0148 4.Rishu Ranjan chowbey 16BEE0090

Elaborate your work status with respect to

i. **Work done after assessment II – Elaborate**

In our project we are converting hand gestures into sign language. For converting we would be using a sensor. Initially we were working with Flex sensors. But after implementing it we summarized that the sensors do not give precise values for the change in other words we were not getting a good amount of sensor output change with a little change in the gesture and thus were not able to differentiate between different gestures.

So the second sensors we finalized for our project is MPU6050 which gives good change in sensor values for a little movement which was the problem with flex sensor. For our final prototype we would be using 5 accelorometers on each finger and setting the range for some particular signs by hit and trial method and accordingly print the gestures and thus bridging the communication gap between dumb person .The code for two gestures that is “Sorry” and “Thank You” has been completed till now which can be seen below.

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
```

```
#endif

MPU6050 mpu;

#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most
boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
//#define count 0
bool blinkState = false;

bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42
bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

  float a,b,c=0.0;
  int count=0;

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and
gravity vector


volatile bool mpuInterrupt = false;     // indicates whether MPU
interrupt pin has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}


void setup() {

   #if I2CDEV_IMPLEMENTATION ==
I2CDEV_ARDUINO_WIRE
       Wire.begin();
```

```
        Wire.setClock(400000);
    #elif I2CDEV_IMPLEMENTATION ==
I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    Serial.begin(115200);
    while (!Serial);

    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);

    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));

    Serial.println(F("\nSend any character to begin DMP programming
and demo: "));
    while (Serial.available() && Serial.read()); // empty buffer
    while (!Serial.available());                 // wait for data
    while (Serial.available() && Serial.read()); // empty buffer again

    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

    if (devStatus == 0) {
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        Serial.print(F("Enabling interrupt detection (Arduino external
interrupt "));
        Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
        Serial.println(F(")..."));
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN),
dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
```

```
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {

        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {

    if (!dmpReady) return;

    while (!mpuInterrupt && fifoCount < packetSize) {
        if (mpuInterrupt && fifoCount < packetSize) {
          fifoCount = mpu.getFIFOCount();
        }
    }

    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    fifoCount = mpu.getFIFOCount();

    if ((mpuIntStatus &
_BV(MPU6050_INTERRUPT_FIFO_OFLOW_BIT)) || fifoCount >=
1024) {
        mpu.resetFIFO();
        fifoCount = mpu.getFIFOCount();
        Serial.println(F("FIFO overflow!"));

    }
```

```
   else if (mpuIntStatus &
_BV(MPU6050_INTERRUPT_DMP_INT_BIT)) {

      while (fifoCount < packetSize) fifoCount =
mpu.getFIFOCount();

      mpu.getFIFOBytes(fifoBuffer, packetSize);
      fifoCount -= packetSize;


      if(count==0){
        delay(7000);
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          mpu.dmpGetGravity(&gravity, &q);
          mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
          a=ypr[0] * 180/M_PI;
          b=ypr[1] * 180/M_PI;
          c=ypr[2] * 180/M_PI;
          Serial.print("b is\t =");
          Serial.print(b);
          Serial.print("c is \t");
          Serial.print(c);
        count=1;
      }

      delay(6000);

      #ifdef OUTPUT_READABLE_YAWPITCHROLL
          mpu.dmpGetQuaternion(&q, fifoBuffer);
          mpu.dmpGetGravity(&gravity, &q);
          mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
          Serial.print("ypr\t");
          Serial.print(ypr[0] * 180/M_PI);
          Serial.print("\t");
          Serial.print(ypr[1] * 180/M_PI);
          Serial.print("\t");
          Serial.println(ypr[2] * 180/M_PI);
      #endif

//Thank you
      if(150 <= (b-ypr[1] * 180/M_PI) <=180 && -100 <= (c-ypr[2] *
180/M_PI) <=-150){
```

```
        Serial.println("Thank you");
      }

//Sorry
      if(-10 <= (b-ypr[1] * 180/M_PI) <=-20 && 72 <= (c-ypr[2] *
180/M_PI) <=78){
        Serial.println("Sorry");
      }



      blinkState = !blinkState;
      digitalWrite(LED_PIN, blinkState);
   }
}
```

We would be incorporating for more gestures and will be using more accelerometers for capturing the movements of all the fingers.

ii. Whether your work has been converted into paper or prototype. If yes, additional weightage will be given.

Yes, our work has been converted into prototype. We are done with testing one sensor for two different words hence 2 different gestures. Calibration of the sensor was done with the code mentioned above. At the moment we have to incorporate more MPU6050 for various other fingers postures for different words.

iii. Component requirements if any

We have ordered 4 more MPU6050 for all the fingers and to also increase the number of gestures and corresponding words.

iv. Difficulty faced if any

Following difficulties were faced-

- Initially we were using flex sensors but were not getting the desired output. We were getting large range of inaccurate values for the slightest bend in the sensors. Now we have finalized MPU6050 for capturing the change.
- Calibrating MPU6050 was done with hit and trial method which was time consuming.

Connection of 5 MPUs with arduino is not possible as we were using I2C communication and MPU6050 has 2 I2C addresses. Later we resolved the problem using a hack mentioned in arduino website.Trick is MPU-6050 sensors are connected at the AD0pins to a separate output of the Arduino. Suppose

all AD0 lines are default high (3.3V), so every MPU-6050 is I2C address 0x69. The Arduino makes one of the AD0 lines low, and uses that sensor at I2C address 0x68. After that is finished, the Arduino selects another AD0 line, and can use that sensor.
So every sensor is used at I2C address 0x68 (one by one) and 0x69 is never used.This will make it possible to have 5 MPU-6050 sensors with one Arduino in our project.