Universitat de Lleida

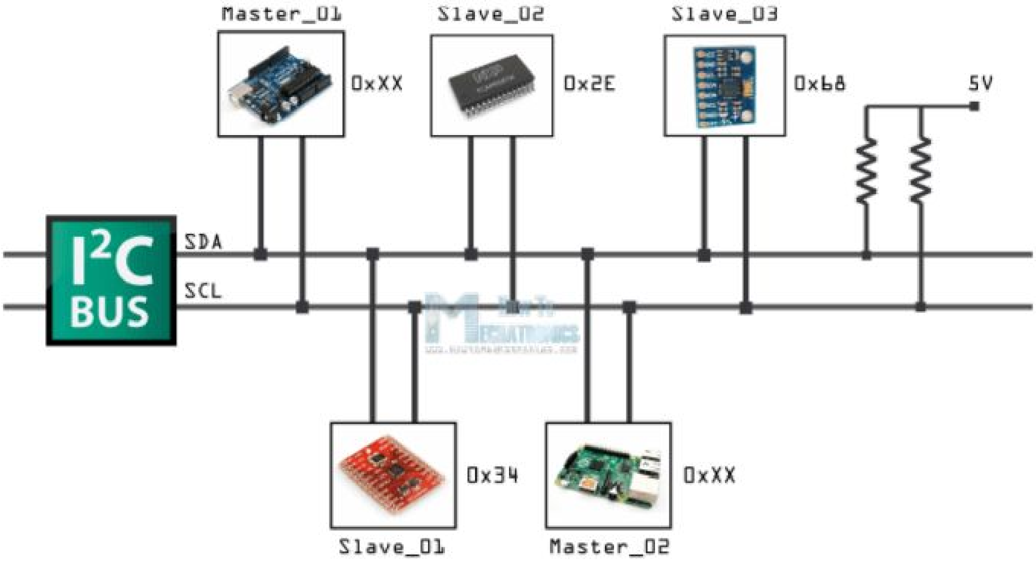
I2c <-> Arduino documentation (Sprint 3)

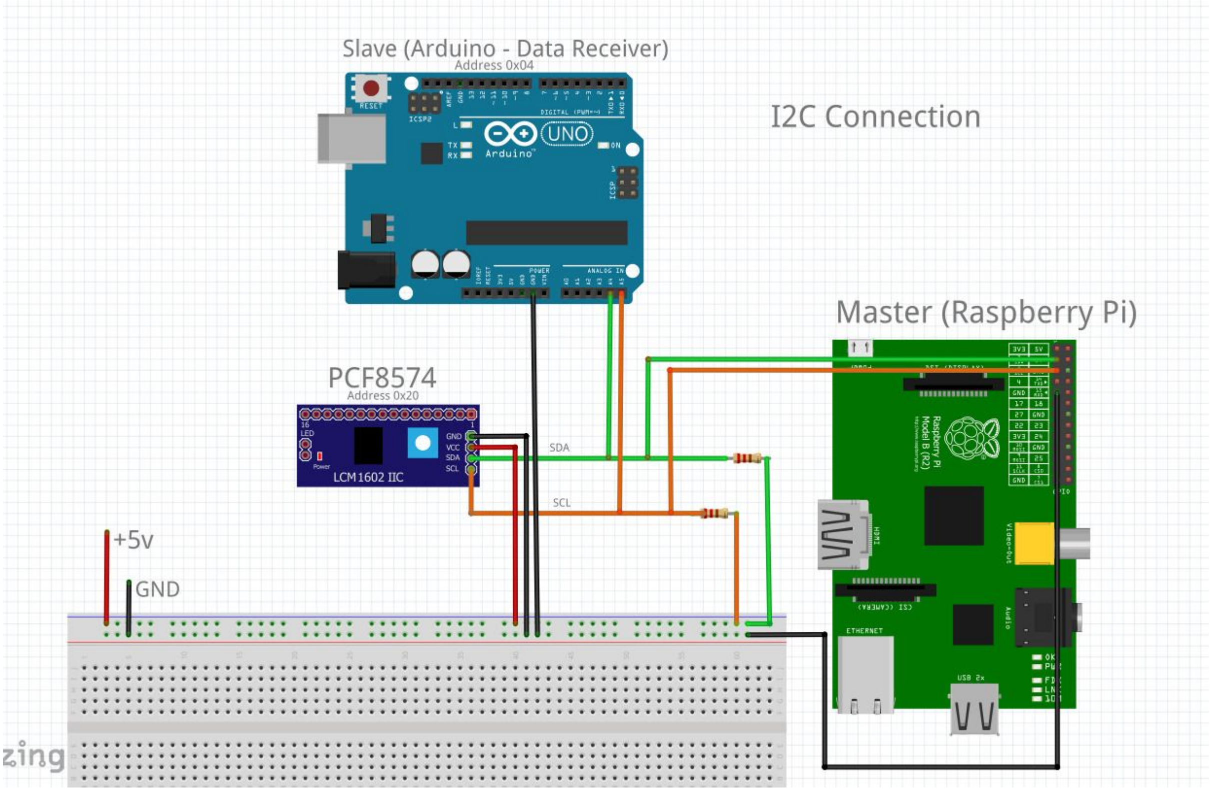
MINF UDL 20-21

Ubiquitous and embedded systems

Team 1

1. **Use Wire Library (Arduino)**
2. **Wiring connection**





Arduino A4 <> SDA

Arduino A5 <> SCL

Raspberry SDA (Pin 2) <> SDA

Raspberry SCL (Pin 3) <> SCL

GND (All) <> GND (All)

3. Code (Arduino)

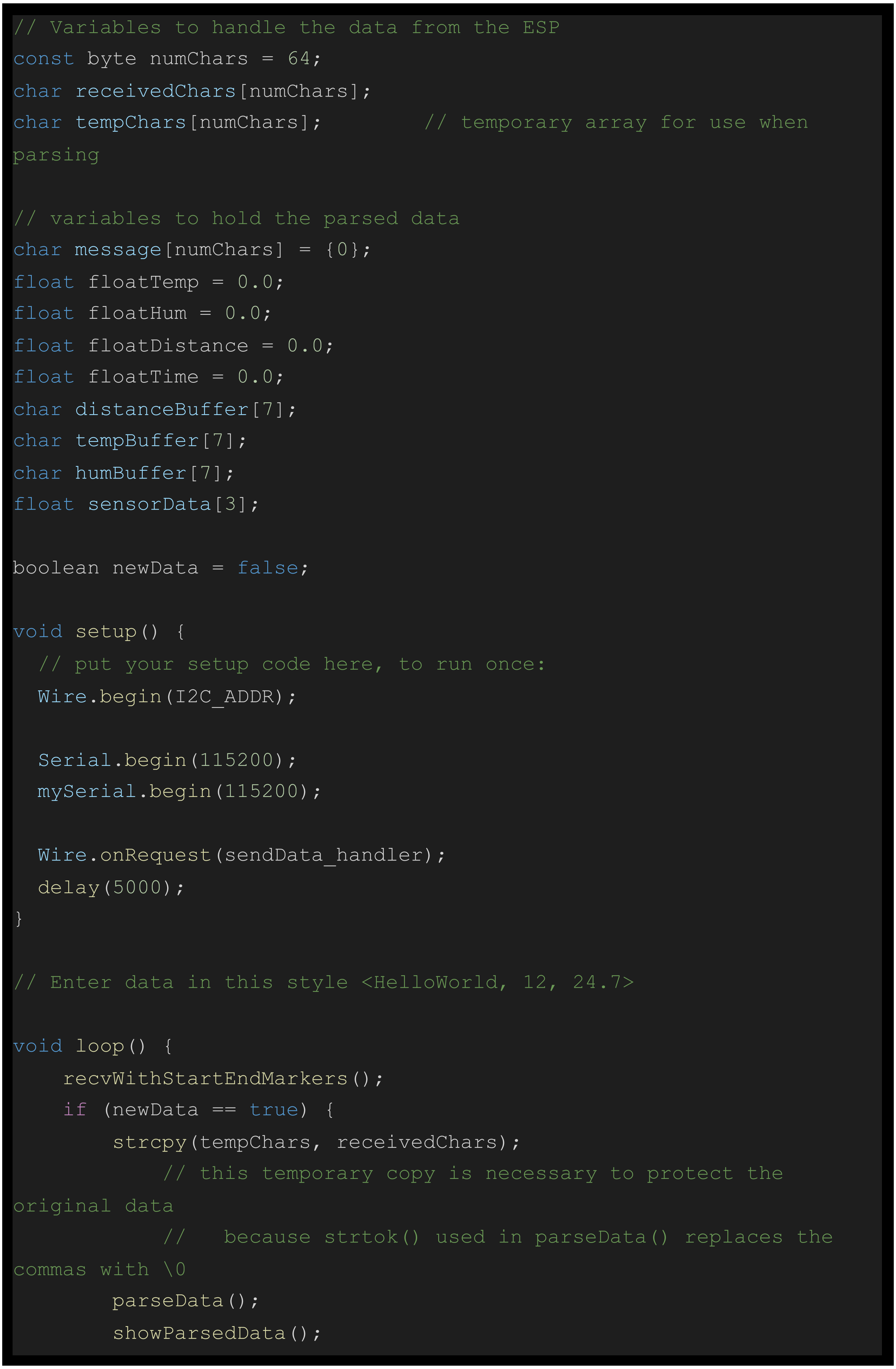
a. Initializing Arduino as a Slave and sending information (3 floats) via the sendHandler function.

b. We initialize the i2c bus with the `Wire.begin(I2C_ADDR);`

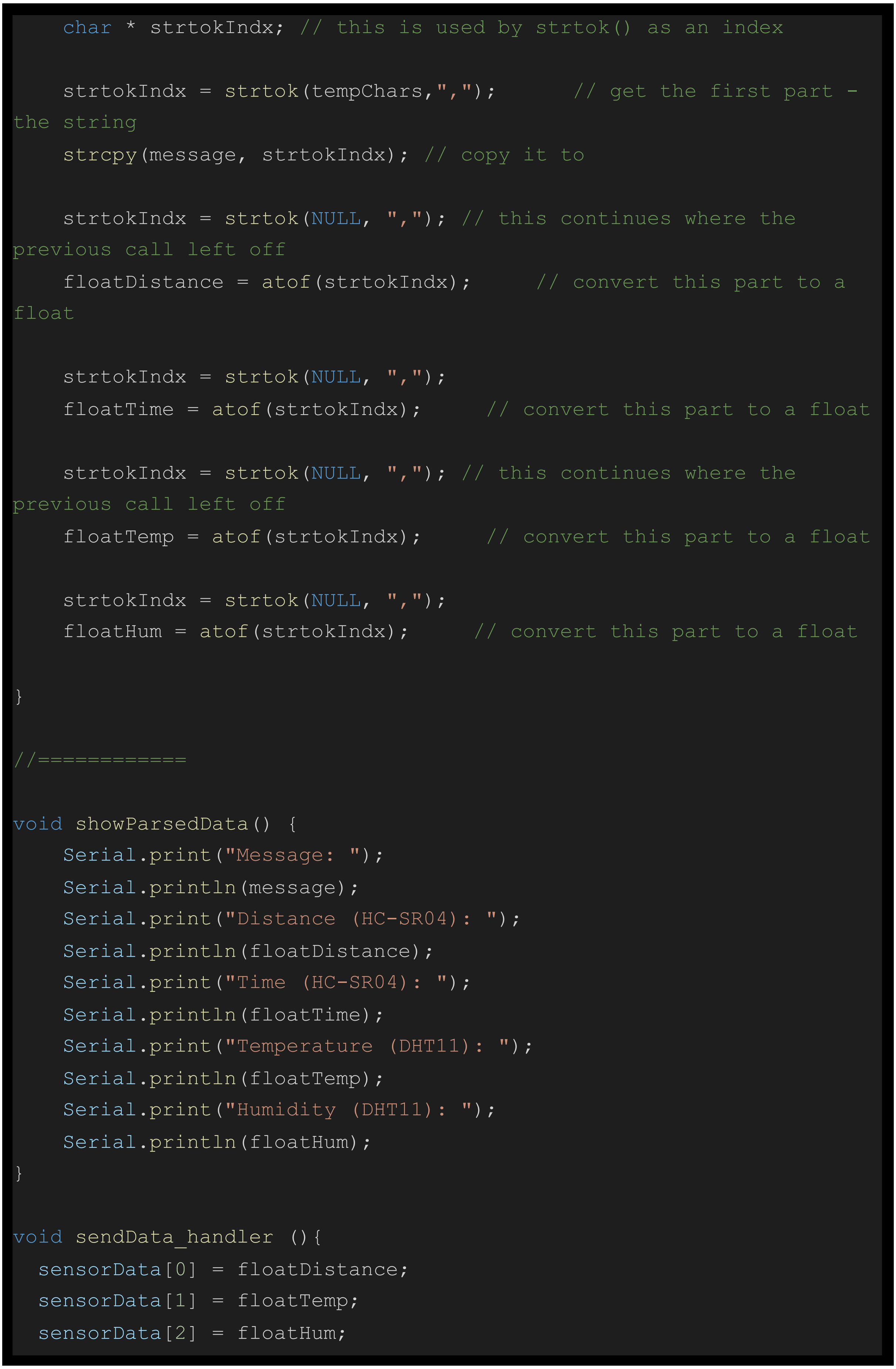
c. Then we answer to the master's i2c request with the function `Wire.onRequest(sendData_handler);`

d. After that the data is sent (collected by the ESP receiver) `Wire.write((byte *) sensorData, sizeof sensorData);`

```
#include <SoftwareSerial.h>
#include <Wire.h>
#define I2C_ADDR 0x04

uint8_t data;

SoftwareSerial mySerial(2, 3); // RX, TX

```



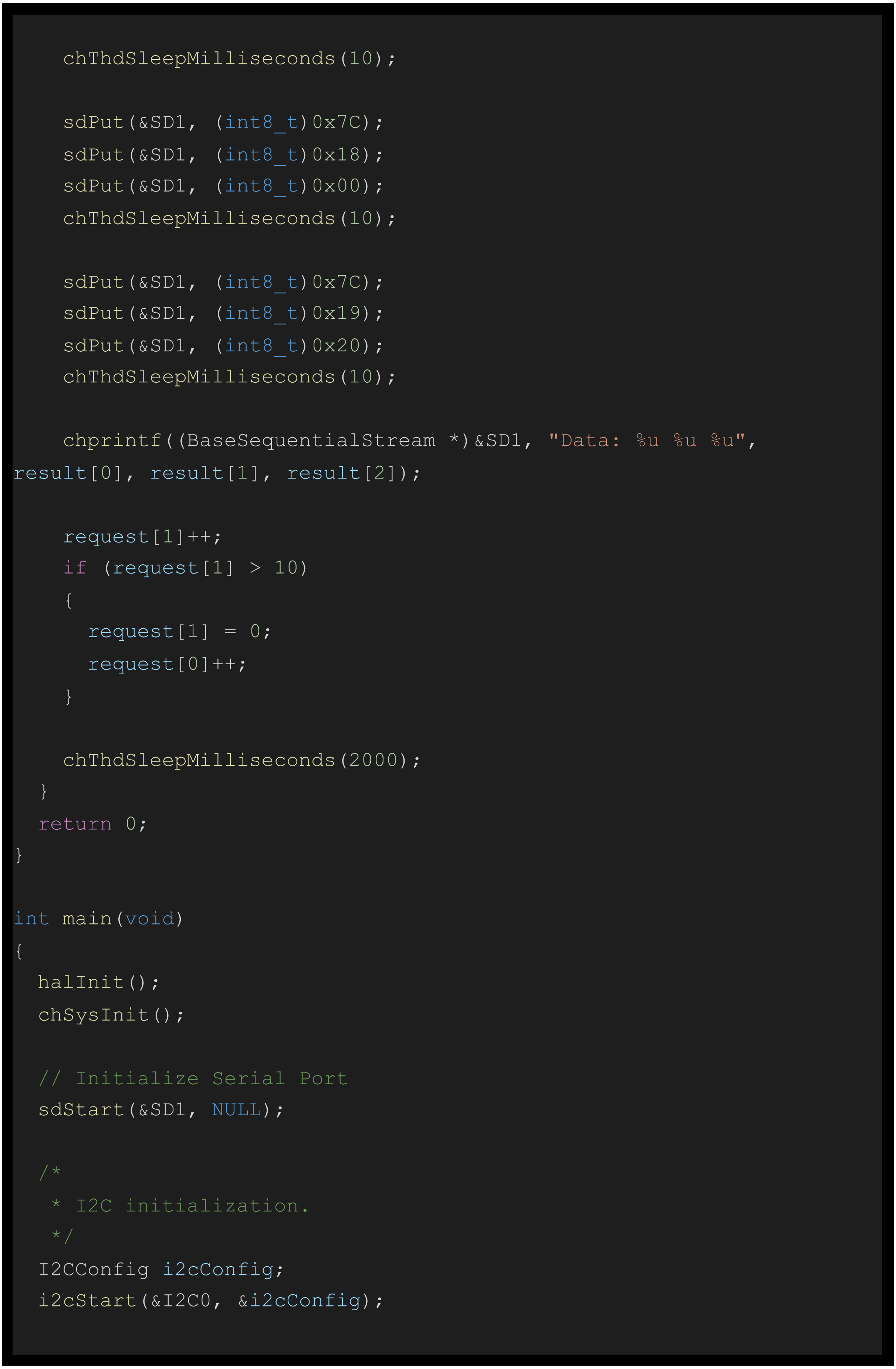
```
// Variables to handle the data from the ESP
const byte numChars = 64;
char receivedChars[numChars];
char tempChars[numChars];        // temporary array for use when
parsing

// variables to hold the parsed data
char message[numChars] = {0};
float floatTemp = 0.0;
float floatHum = 0.0;
float floatDistance = 0.0;
float floatTime = 0.0;
char distanceBuffer[7];
char tempBuffer[7];
char humBuffer[7];
float sensorData[3];

boolean newData = false;

void setup() {
  // put your setup code here, to run once:
  Wire.begin(I2C_ADDR);

  Serial.begin(115200);
  mySerial.begin(115200);

  Wire.onRequest(sendData_handler);
  delay(5000);
}

// Enter data in this style <HelloWorld, 12, 24.7>

void loop() {
    recvWithStartEndMarkers();
    if (newData == true) {
        strcpy(tempChars, receivedChars);
            // this temporary copy is necessary to protect the
original data
            //   because strtok() used in parseData() replaces the
commas with \0
        parseData();
        showParsedData();
        newData = false;
    }
}

//============

void recvWithStartEndMarkers() {
    static boolean recvInProgress = false;
    static byte ndx = 0;
    char startMarker = '<';
    char endMarker = '>';
    char rc;

    while (mySerial.available() > 0 && newData == false) {
        rc = mySerial.read();

        if (recvInProgress == true) {
            if (rc != endMarker) {
                receivedChars[ndx] = rc;
                ndx++;
                if (ndx >= numChars) {
                    ndx = numChars - 1;
                }
            }
            else {
                receivedChars[ndx] = '\0'; // terminate the string
                recvInProgress = false;
                ndx = 0;
                newData = true;
            }
        }

        else if (rc == startMarker) {
            recvInProgress = true;
        }
    }
}

//============

void parseData() {      // split the data into its parts
```



```
    char * strtokIndx; // this is used by strtok() as an index

    strtokIndx = strtok(tempChars,",");      // get the first part -
the string
    strcpy(message, strtokIndx); // copy it to

    strtokIndx = strtok(NULL, ","); // this continues where the
previous call left off
    floatDistance = atof(strtokIndx);     // convert this part to a
float

    strtokIndx = strtok(NULL, ",");
    floatTime = atof(strtokIndx);     // convert this part to a float

    strtokIndx = strtok(NULL, ","); // this continues where the
previous call left off
    floatTemp = atof(strtokIndx);     // convert this part to a float

    strtokIndx = strtok(NULL, ",");
    floatHum = atof(strtokIndx);     // convert this part to a float

}

//============

void showParsedData() {
    Serial.print("Message: ");
    Serial.println(message);
    Serial.print("Distance (HC-SR04): ");
    Serial.println(floatDistance);
    Serial.print("Time (HC-SR04): ");
    Serial.println(floatTime);
    Serial.print("Temperature (DHT11): ");
    Serial.println(floatTemp);
    Serial.print("Humidity (DHT11): ");
    Serial.println(floatHum);
}

void sendData_handler (){
  sensorData[0] = floatDistance;
  sensorData[1] = floatTemp;
  sensorData[2] = floatHum;
  Wire.write((byte *) sensorData, sizeof sensorData);

  delay(100);

}
```

4. Code (Raspberry)

```
#include "ch.h"
#include "hal.h"
#include "chprintf.h"

static const uint8_t arduino_address = 0x04;
static const uint8_t pcf_address = 0x27;

static WORKING_AREA(waThread_I2C, 128);
static msg_t Thread_I2C(void *p)
{
  (void)p;
  chRegSetThreadName("SerialPrintI2C");
  uint8_t request[] = {0, 0};
  uint8_t result[3];
  uint8_t aux;
  uint8_t bit = 0b10000000;

  msg_t status;

  // Some time to allow slaves initialization
  chThdSleepMilliseconds(2000);

  while (TRUE)
  {

    // Request values
    i2cMasterTransmit(&I2C0, arduino_address, request, 2,
                      &result, 3);

    // Attempt to send the bits to the pcf
    i2cMasterTransmit(&I2C0, pcf_address, &bit, sizeof(bit),
                      &aux, 0);
```



```
    chThdSleepMilliseconds(10);

    sdPut(&SD1, (int8_t)0x7C);
    sdPut(&SD1, (int8_t)0x18);
    sdPut(&SD1, (int8_t)0x00);
    chThdSleepMilliseconds(10);

    sdPut(&SD1, (int8_t)0x7C);
    sdPut(&SD1, (int8_t)0x19);
    sdPut(&SD1, (int8_t)0x20);
    chThdSleepMilliseconds(10);

    chprintf((BaseSequentialStream *)&SD1, "Data: %u %u %u",
result[0], result[1], result[2]);

    request[1]++;
    if (request[1] > 10)
    {
      request[1] = 0;
      request[0]++;
    }

    chThdSleepMilliseconds(2000);
  }
  return 0;
}

int main(void)
{
  halInit();
  chSysInit();

  // Initialize Serial Port
  sdStart(&SD1, NULL);

  /*
   * I2C initialization.
   */
  I2CConfig i2cConfig;
  i2cStart(&I2C0, &i2cConfig);
  chThdCreateStatic(waThread_I2C, sizeof(waThread_I2C),
                    HIGHPRIO, Thread_I2C, NULL);

  // Blocks until finish
  chThdWait(chThdSelf());

  return 0;
}
```

5. Considerations:
    a.