Mechatronics Project Documentation

Avni Gupta, Amiruthavallii Karthi Kumar, Aasimah Shaheen, Adiba Tahsin

6CCYB080 Mechatronics

TABLE OF CONTENTS

1. PURPOSE OF THE SYSTEM

The aim of this project was to create a wearable pulse oximeter to measure and monitor heart rate and blood oxygen levels. To do this, components such as a pulse oximetry sensor (Gravity MAX30102 sensor), a microcontroller board (Arduino Nano ESP32), and a Mini Breadboard were engineered to serve as a wearable device.

The purpose of this technology is to provide users with real-time health data, enabling them to track vital signs seamlessly. The Gravity MAX30102 sensor, when positioned on the fingertip, employs advanced optical technology to non-invasively detect the user's pulse and measure blood oxygen saturation levels.

The Arduino Nano ESP32 processes the raw data from the MAX30102 sensor and transmits it wirelessly to a dedicated app via Bluetooth or Wi-Fi. The app serves as an intuitive interface, presenting the data in a user-friendly manner and offering insights into the wearer's cardiovascular health. The Mini Breadboard serves as a convenient platform for connecting and organising the electronic components of the system.

The potential applications of this device are vast, ranging from fitness monitoring during workouts, to medical use for individuals with specific health conditions such as heart arrhythmia. The integration of this wearable technology and the Arduino Cloud Internet of Things (IoT) provides a way for the user to access historical data allowing individuals to monitor their health proactively. It offers a non-intrusive and efficient way to monitor key physiological parameters, empowering users to make informed decisions about their well-being. This means that users can now monitor their vital signs on the go or from the comfort of their homes, minimising hospital visits and the burden on clinical facilities.

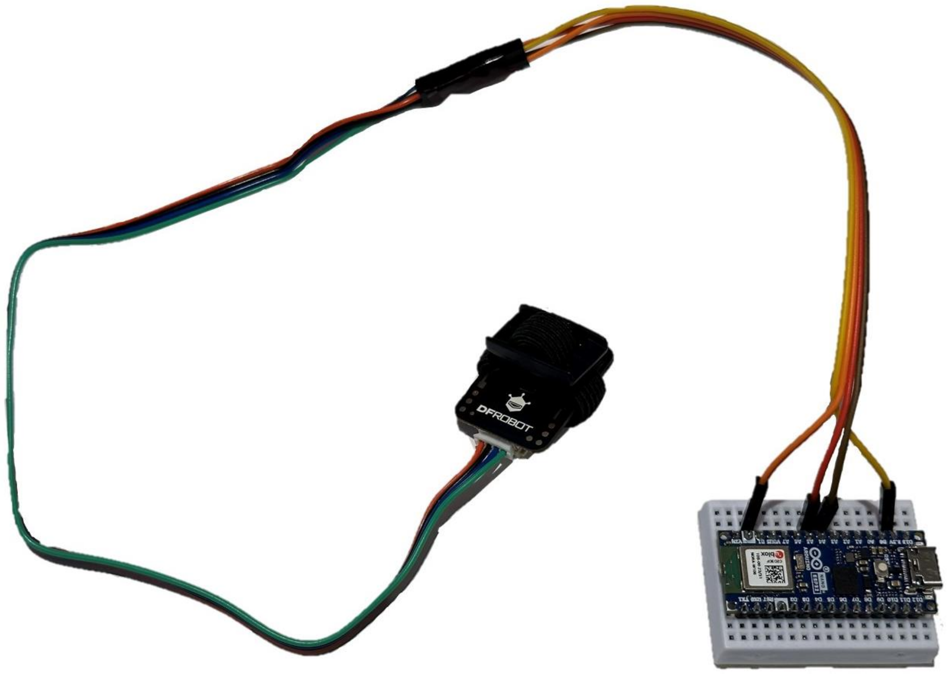


Figure 1: Arduino-based Pulse Oximeter

2. COMPONENTS, SUPPLIERS & COSTS

Gravity: MAX30102 Heart Rate and Oximeter Sensor

Suggested supplier: The PiHut - £21.30

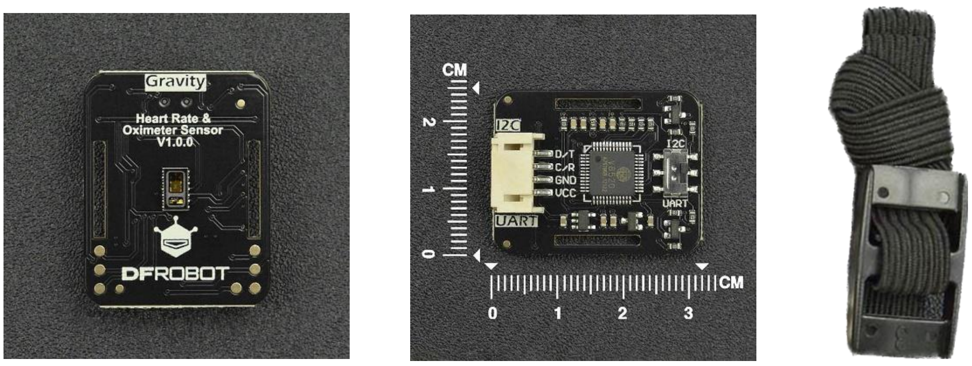


Figure 2: Gravity MAX30102 by DFROBOT and adjustable finger strap [1]

MAX30102 is a versatile sensor measuring both blood oxygen concentration and the heart rate of the user through the Maxim MAX30102 chip. It uses PPG (Photoplethysmography) to measure data, and then outputs through I2C pins. As an added safety feature, the temperature of the board is also measured to ensure the monitor works at its optimum performance whilst being safe for the user during physical contact.

The provided elastic attachment is used to connect the user's finger to the sensor for a more accurate measurement as varying pressure can affect the measurement accuracy. It can be adjusted to be made larger or smaller based on the requirements of the user.

Male-to-female jumper wires

Suggested supplier: ELEGOO on Amazon.co.uk - £6.99

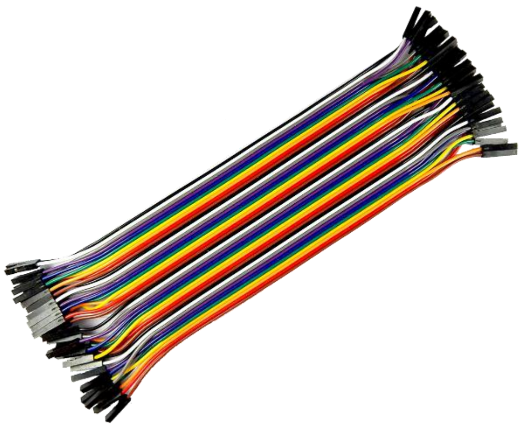


Figure 3: ELEGOO Male-to-female Jumper wires for microcontroller to sensor connection

Arduino Nano ESP32 (with headers)

Suggested supplier: The PiHut - £19.80

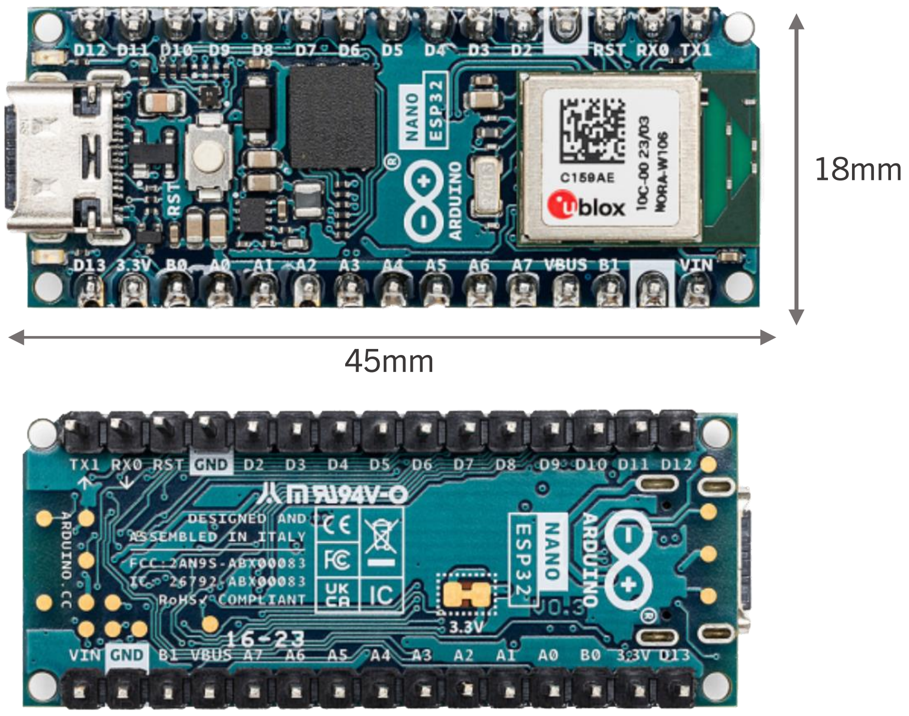


Figure 4: Arduino Nano ESP32 | Compatible with Arduino Cloud [2]

Supply voltage: 5V by USB

Digital pins operating voltage: 3.3V

The Nano ESP32 is a small but powerful microcontroller making it very suitable for embedding in wearable projects such as this pulse oximeter. It includes 8 analogue input pins which is sufficient for this device and can use I2C communication to communicate with the MAX30102 sensor. This module supports both Wi-Fi and Bluetooth such that it enables Arduino IoT Cloud development.

USB-C cable

Suggested supplier: Amazon Basics on Amazon.co.uk - £7.10

Figure 5: USB-C Cable to connect power supply to Arduino Nano ESP32

Mini Breadboard

Suggested supplier: The PiHut - £1.50

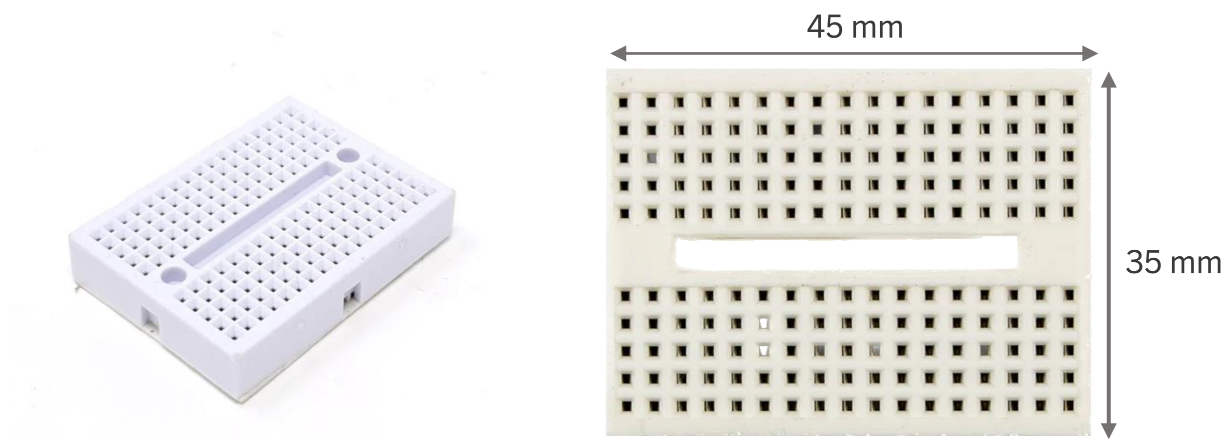


Figure 6: Mini breadboard to allow connections between the Arduino Nano ESP32 and the MAX30102 sensor [3]

This mini breadboard is compact yet big enough to hold the nano esp32. This makes it easy to connect the sensor to the required pins. It also comes with a sticky back so it could be stuck to the casing/arm band to make the device wearable.

Arduino Cloud App

Suggested Supplier: App Store or https://cloud.arduino.cc/ - free

The sensor readings are displayed on the Arduino Cloud dashboard which can be accessed either via the Arduino IoT Cloud Remote app or using the Arduino Cloud desktop site. This allows multiple people to track the readings as opposed to if an LED screen were used where only the user is able to see the values. For example, healthcare professionals can track the vital signs of their patients at home.

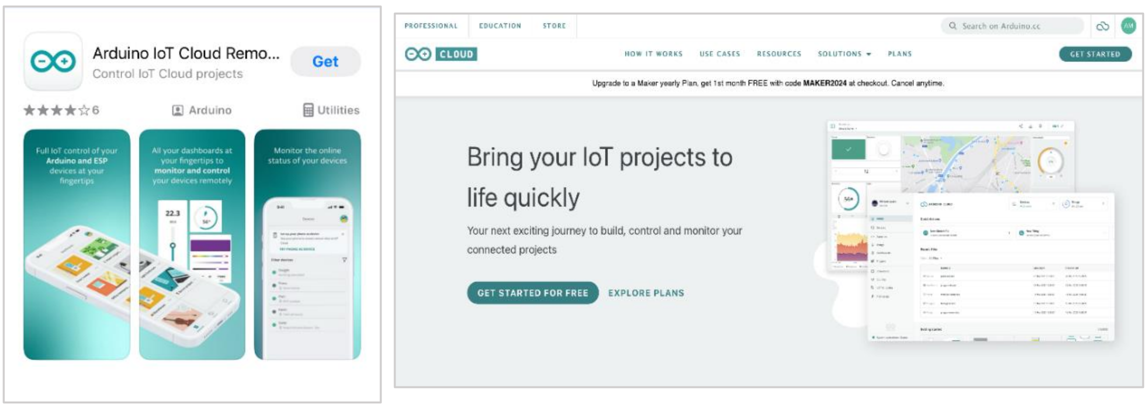

Arduino IoT Cloud Remo...
Control IoT Cloud projects
Get
Arduino
Utilities
Full IoT control of your Arduino and ESP devices at your fingertips
All your dashboards at your fingertips to monitor and control your devices remotely
Monitor the online status of your devices
PROFESSIONAL
EDUCATION
STORE
CLOUD
HOW IT WORKS
USE CASES
RESOURCES
SOLUTIONS
PLANS
GET STARTED
Upgrade to a Maker yearly Plan, get 1st month FREE with code MAKER2024 at checkout. Cancel anytime.
Bring your IoT projects to life quickly
Your next exciting journey to build, control and monitor your connected projects
GET STARTED FOR FREE
EXPLORE PLANS

Figure 7: Arduino Cloud IoT Remote Mobile app and Desktop site

Armband

Suggested supplier: Amazon - £11.99

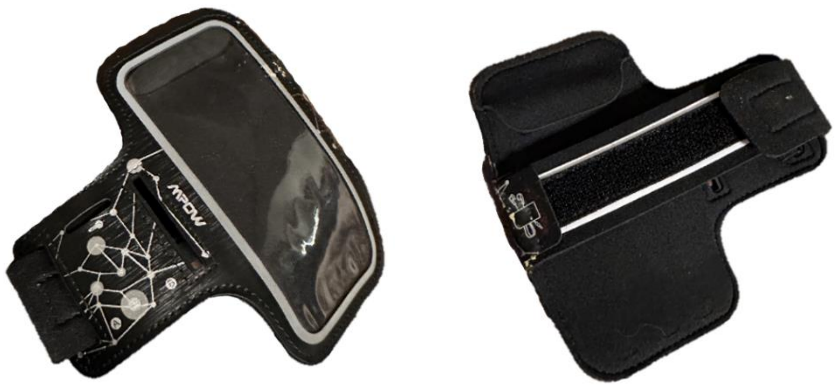
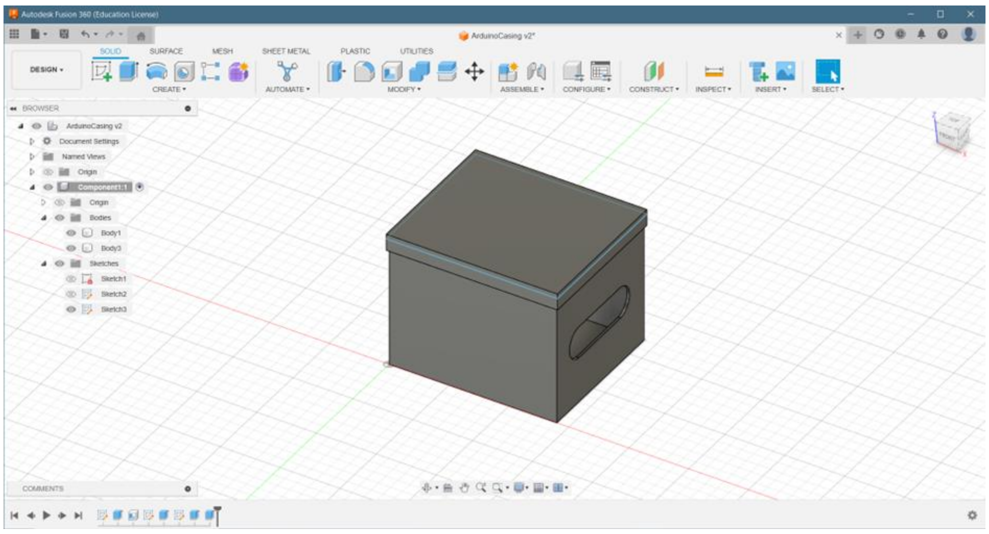


Figure 8: Arm band to attach pulse oximeter device to user

This is easy to wear, and the adjustable arm band can be fastened to the user's arm. It contains the microcontroller as well as the powering unit. This makes the device wearable and portable.

Casing

Suggested supplier: Fusion 360 – subscription, FLSUN QQ-S printer - £349, 1.75mm PLA 3D filament (Cold white-1kg) - £20.99



This is a casing which contains the main components of the device including the breadboard, microcontroller, and the wiring. It provides a seamless, clean finish to the overall design of the device.

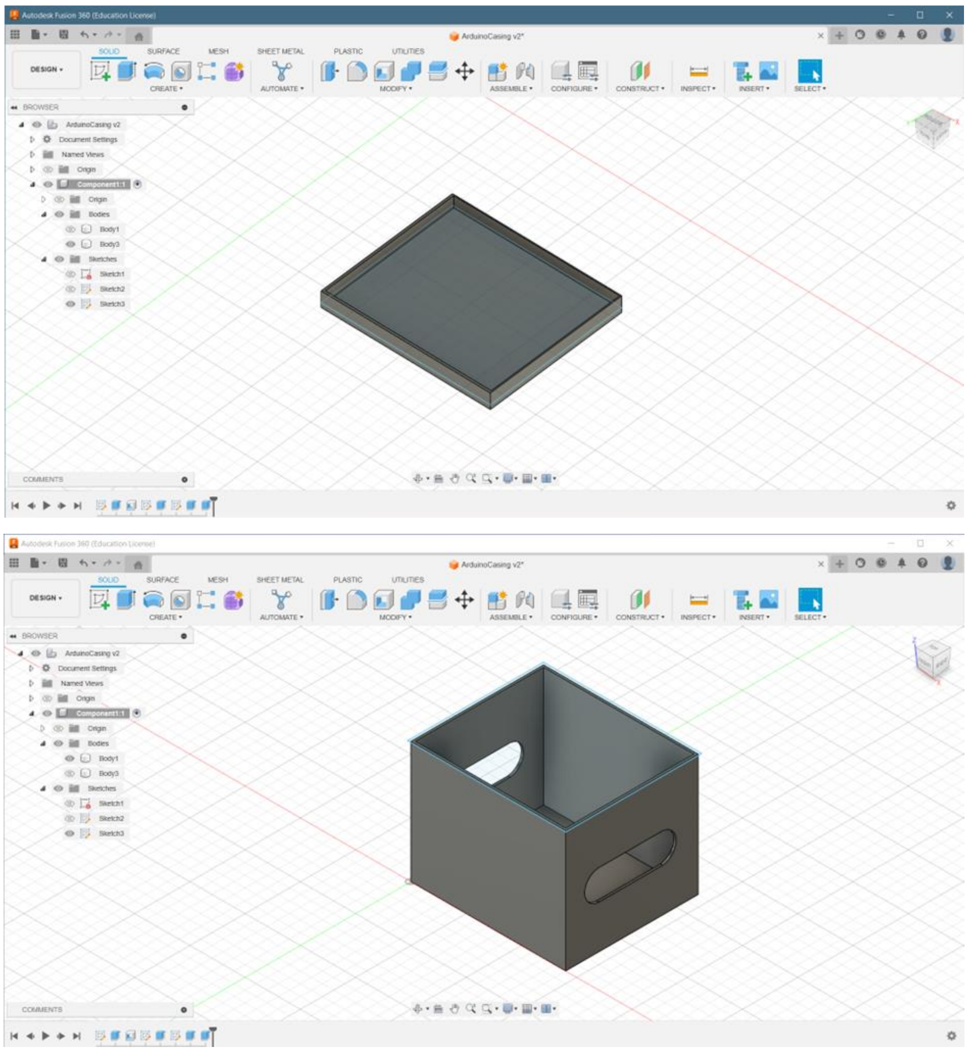


Figure 9: Design of casing on Autodesk Fusion 360

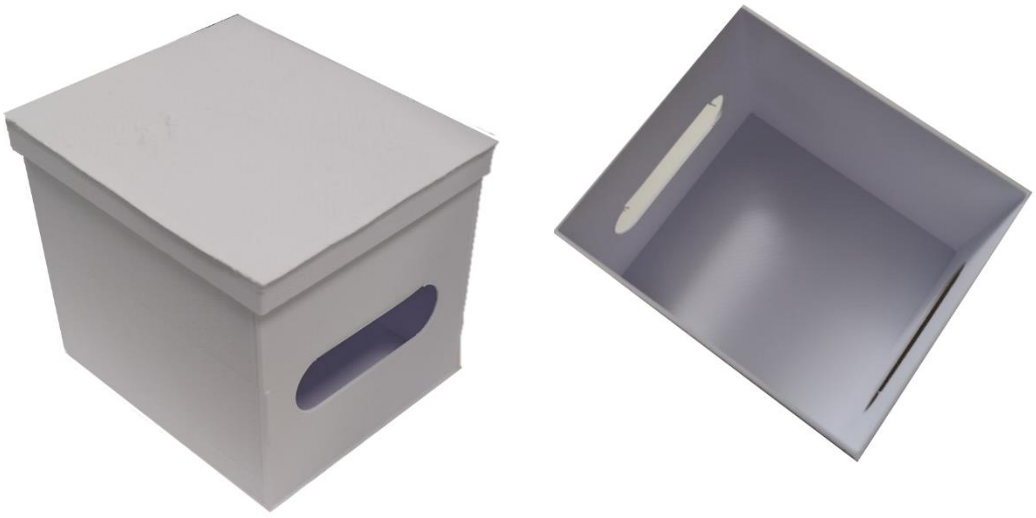


Figure 10: 3D printed casing

3. BLOCK DIAGRAM

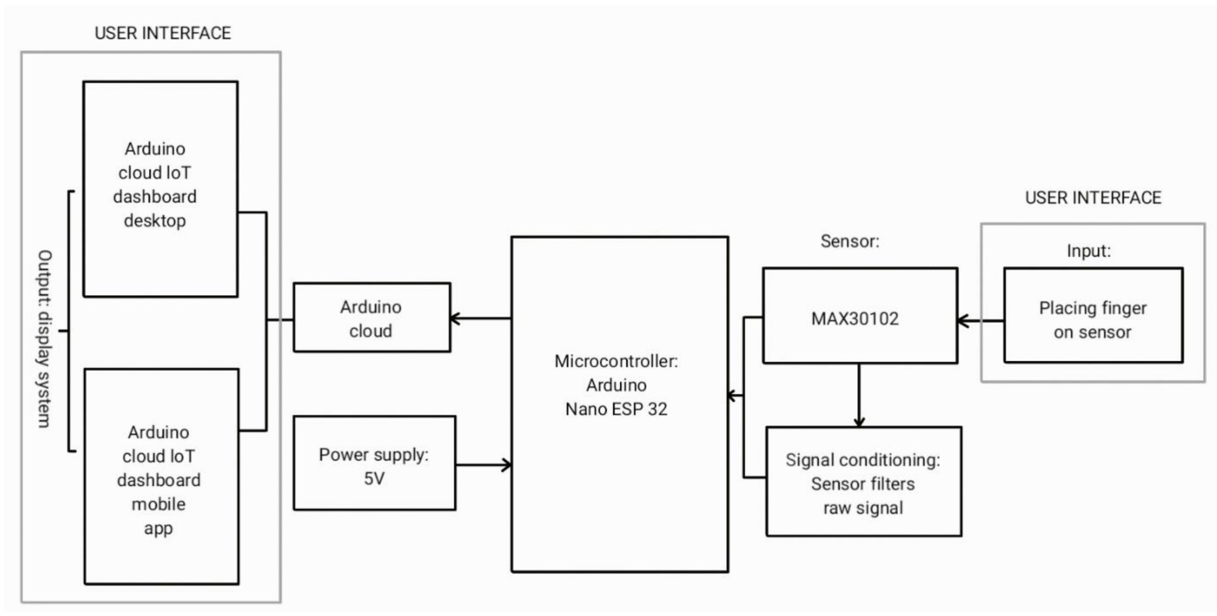

USER INTERFACE
Output: display system
Arduino cloud IoT dashboard desktop
Arduino cloud IoT dashboard mobile app
Arduino cloud
Power supply: 5V
Microcontroller: Arduino Nano ESP 32
Sensor:
MAX30102
Signal conditioning: Sensor filters raw signal
USER INTERFACE
Input:
Placing finger on sensor

Figure 11: Block diagram of system

System operations

The MAX30102 heart rate and oxygen sensor is based on the principle of light absorption by blood. It uses red-light reflection and infrared technology to measure the saturation of oxygen and heart rate of the user [4]. The amount of light absorbed is proportional to the concentration of oxygenated haemoglobin. This method is called Photoplethysmogram (PPG) [5]. PPG also measures volumetric changes in blood circulation. This produces waveforms which can be interpreted by the sensor's own microcontroller unit to calculate heart rate. It receives the signal and converts it into analogue data. It is then converted into digital signals using an analogue-to-digital converter (ADC) which is already built into the sensor.

The sensor then uses a range of digital signal processing algorithms to filter and process the raw PPG data. These signal conditioned readings are then outputted through the I2C, used for communication with the Arduino Nano [5]. The code used for interfacing the MAX30102 on an Arduino Nano involves many tasks including:

- Downloading and installing the MAX30102 Arduino library.
- Include the library in your Arduino sketch.
- Use the library functions to read sensor data and send it to Arduino Cloud using a Wi-Fi connection.
- Download the Arduino Cloud IoT app on a mobile phone and configure it to display data.

4. CIRCUIT DIAGRAM

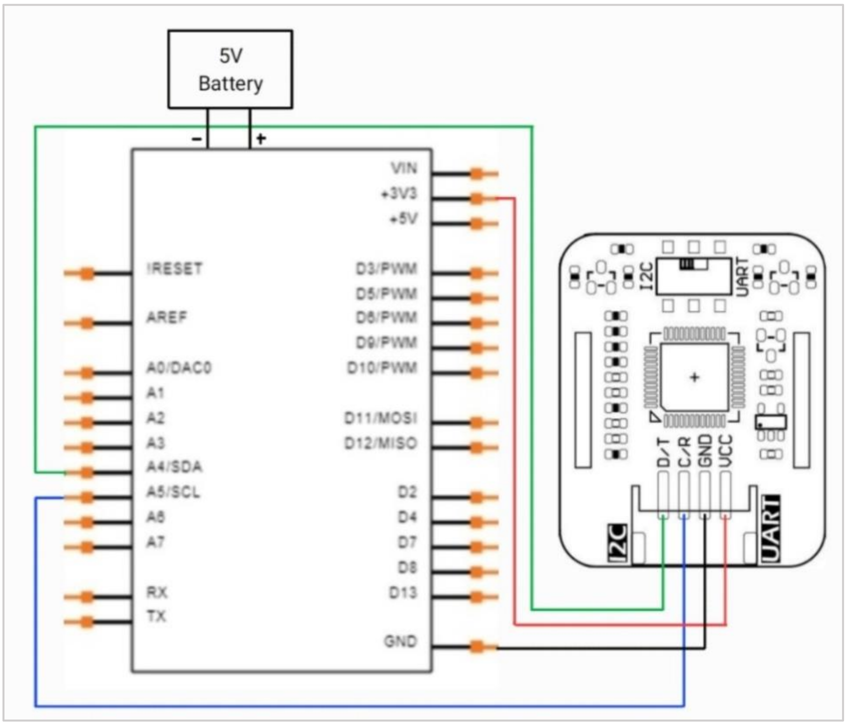


Figure 12: Circuit diagram of system

5. WIRING DIAGRAM

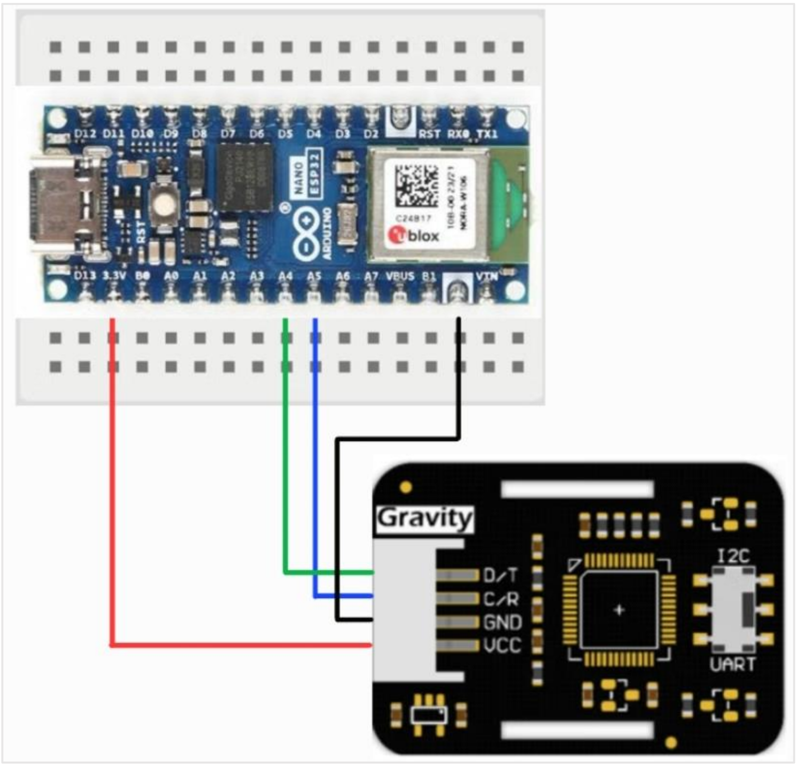
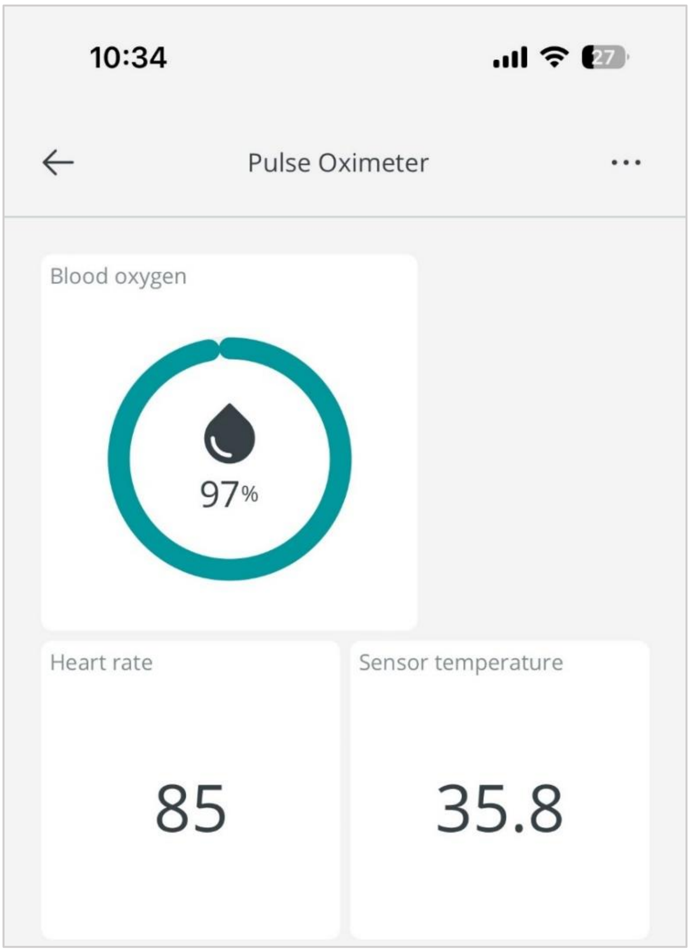

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 RST RX0 TX1
D13 3.3V B0 A0 A1 A2 A3 A4 A5 A6 A7 VBUS B1 VIN
NANO ESP32
ARDUINO
blox
Gravity
D/T
C/R
GND
VCC
I2C
UART

Figure 13: Wiring diagram of system

6. INSTRUCTIONS FOR USE OF SYSTEM

1. When activated, the sensor will be illuminated with a red light.
2. To operate, simply place index finger on the light and secure with the elastic attachment.
3. Wait for 5 seconds to allow the sensor to calibrate and readings to show up.
4. Use the app to view the dashboard with the values.

Figure 14: Depiction of user placing fingertip on sensor



Blood oxygen

97%

Heart rate

85

Sensor temperature

35.8

Figure 15: Arduino Cloud IoT remote app dashboard showing widgets with live sensor readings

7. EXAMPLE GRAPHS OF CAPTURED DATA

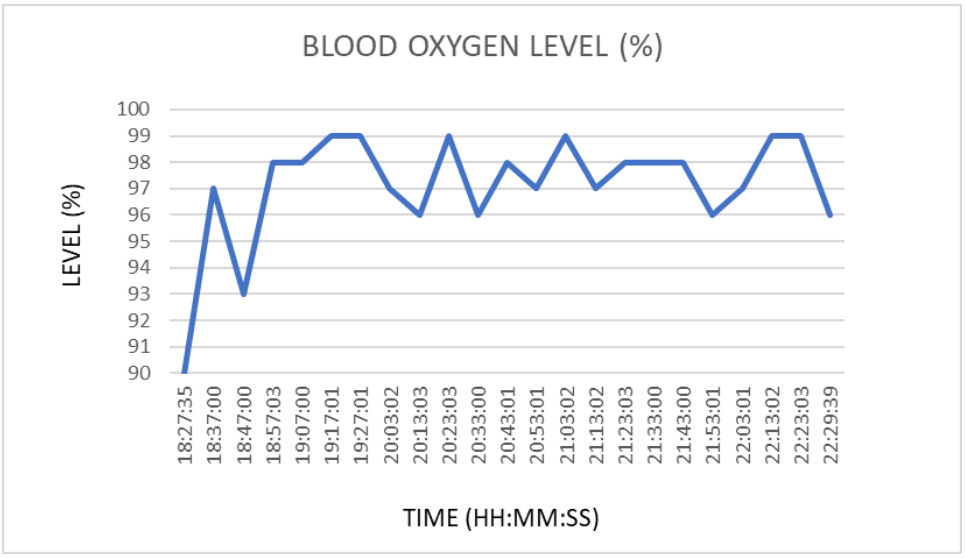


Figure 16: Graph of blood oxygen levels plotted against timestamp

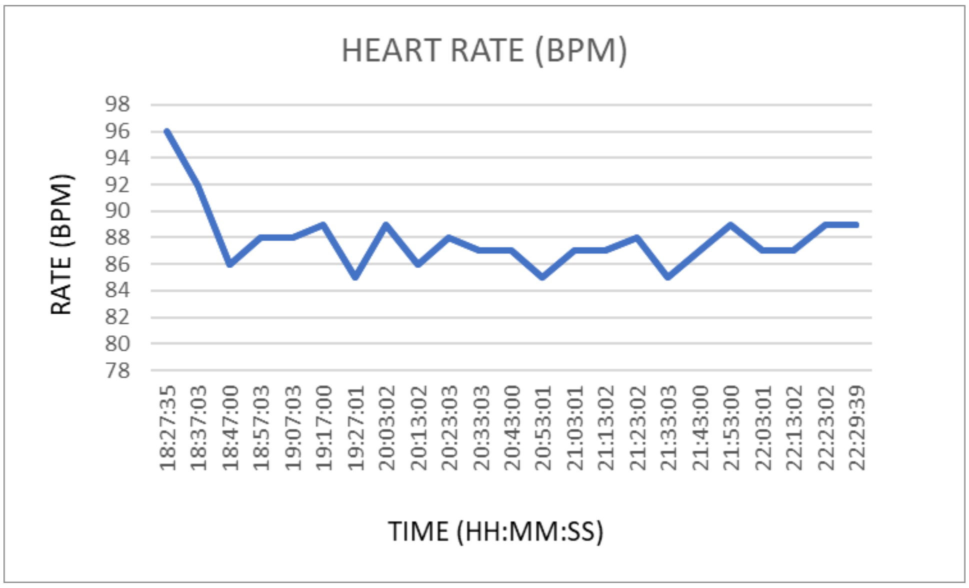


Figure 17: Graph of heart rate plotted against timestamp

Note: *the data in the excel files have been sampled to produce the graphs in figures 16 and 17 as the large number of datapoints collected over a period of multiple hours did not facilitate a good graphical visualisation. Please view excel files attached in project submission for the full data.*

8. INCLUSIVE DESIGN AND EXTRA FEATURES

Inclusive design

1. Adjustable Elastic Attachment: MAX30102's elastic attachment caters to various finger sizes, promoting inclusivity.

2. Intuitive Operation: The system's simple operation ensures accessibility for users with diverse technical proficiencies.

3. Visual Feedback: colour-coded dashboard provides clear visual feedback, aiding users, including those with visual impairments.

Extra features

1. Temperature of the sensor chip as a safety mechanism
2. Arduino cloud IoT remote app for the user
3. Wearable (3D printed casing to contain the microcontroller, arm band to attach to user, elastic band to fasten sensor to fingertip)
4. Threshold for low blood oxygen levels – dashboard turns red if levels go lower than 95%

9. SOURCE CODE

Arduino IDE code [6]

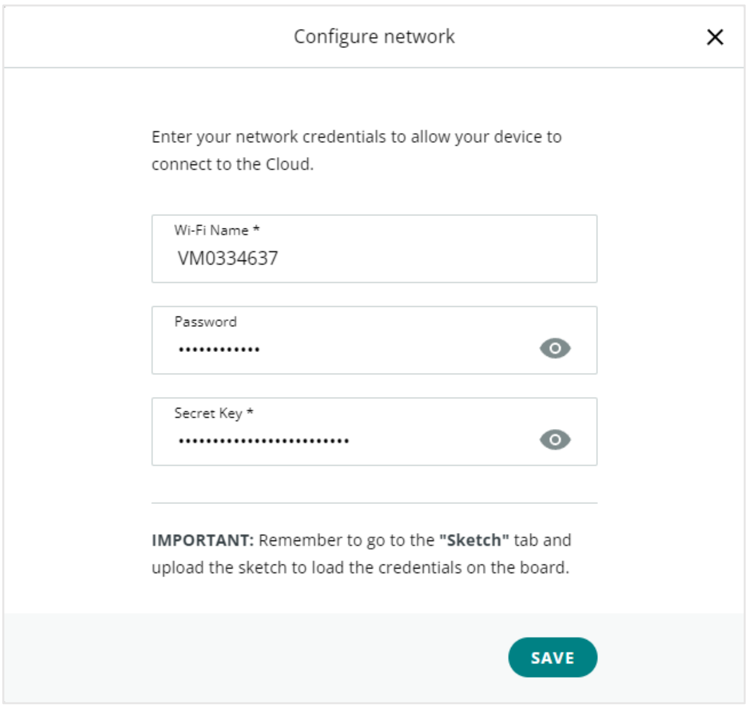
```
// Include Sensor library for built-in functions
#include "DFRobot_BloodOxygen_S.h"

// Communication configuration
#define I2C_COMMUNICATION
#ifdef  I2C_COMMUNICATION

// Define I2C address of sensor
#define I2C_ADDRESS    0x57

// Create I2C instance for communication
  DFRobot_BloodOxygen_S_I2C MAX30102(&Wire ,I2C_ADDRESS);

#else // Fallback to serial communication if I2C is unavailable

#if defined(ARDUINO_AVR_UNO) || defined(ESP8266)
SoftwareSerial mySerial(4, 5);
DFRobot_BloodOxygen_S_SoftWareUart MAX30102(&mySerial, 9600);
#else
```

```
// Use hardware serial port
DFRobot_BloodOxygen_S_HardWareUart MAX30102(&Serial1, 9600);
#endif

#endif

void setup()
{
  Serial.begin(115200); // Start serial communication for debugging

  // Initialise the MAX30102 sensor
  while (false == MAX30102.begin())
  {
    Serial.println("Sensor initialisation failed. Retrying…");
    delay(1000);
  }
  Serial.println("Sensor initialisation successful!");
  Serial.println("Starting measuring...");

  // Start collecting sensor data
  MAX30102.sensorStartCollect();
}

void loop()
{

  // Get heart rate and blood oxygen data from MAX30102 sensor
  MAX30102.getHeartbeatSPO2();
  Serial.print("SPO2 is : ");
  Serial.print(MAX30102._sHeartbeatSPO2.SPO2);
  Serial.println("%");
  Serial.print("heart rate is : ");
  Serial.print(MAX30102._sHeartbeatSPO2.Heartbeat);
  Serial.println("Times/min");
  Serial.print("Temperature value of the board is : ");
  Serial.print(MAX30102.getTemperature_C());
  Serial.println(" ℃");

  // The sensor updates the data every 4 seconds
  delay(4000);

  // Uncomment the lines below to stop measuring
  // Serial.println("stop measuring...");
  // MAX30102.sensorEndCollect();
}
```

Arduino Cloud code

```
// Sensor library for functionality
#include <DFRobot_BloodOxygen_S.h>

/*
  Sketch generated by the Arduino IoT Cloud Thing "Pulse Oximeter"
  https://create.arduino.cc/cloud/things/364ddafc-8c4c-4cba-bf5b-c29bac8577ee

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes
are made to the Thing

  float temperature;
  int blood_oxygen;
  int heart_rate;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this
sketch.
*/

// Automatically generated variables and functions for Arduino IoT Cloud
#include "thingProperties.h"

// Communication configuration (prioritise I2C)
#define I2C_COMMUNICATION

#ifdef I2C_COMMUNICATION
#define I2C_ADDRESS 0x57 // I2C address of the sensor
DFRobot_BloodOxygen_S_I2C MAX30102(&Wire, I2C_ADDRESS); // I2C instance
#else

// Fallback to serial communication if I2C is unavailable
#if defined(ARDUINO_AVR_UNO) || defined(ESP8266)
SoftwareSerial mySerial(4, 5);
DFRobot_BloodOxygen_S_SoftWareUart MAX30102(&mySerial, 9600);
#else
DFRobot_BloodOxygen_S_HardWareUart MAX30102(&Serial1, 9600);
#endif
#endif

void setup() {

  // Initialize serial and wait for port to open
  Serial.begin(115200);
  // This delay gives the chance to wait for a Serial Monitor without blocking
if none is found
  delay(1000);

  // Initialise Arduino IoT Cloud variables and properties
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you’ll get.
     The default is 0 (only errors).
     Maximum is 4
  */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  // Sensor starts collecting data
  MAX30102.sensorStartCollect();
}

void loop() {
  // Read sensor data and update cloud variables

  sensor();
  ArduinoCloud.update();
}

void sensor() {

  // Get heart rate and blood oxygen data
  MAX30102.getHeartbeatSPO2();

  // Heart rate
  heart_rate = MAX30102._sHeartbeatSPO2.Heartbeat;
  Serial.print("heart rate is : ");
  Serial.print(heart_rate);
  Serial.println("Times/min");


  // Blood oxygen
  blood_oxygen = MAX30102._sHeartbeatSPO2.SPO2;
  Serial.print("SPO2 is : ");
  Serial.print(blood_oxygen);
  Serial.println("%");
  // Temperature of the sensor
  temperature = MAX30102.getTemperature_C();
  Serial.print("Temperature value of the board is : ");
  Serial.print(temperature);
  Serial.println(" °C");

  // delay(4000);
}
```

Arduino Nano ESP32 Wi-Fi Configuration on Arduino Cloud


Configure network

Enter your network credentials to allow your device to connect to the Cloud.

Wi-Fi Name *
VM0334637

Password
••••••••••••

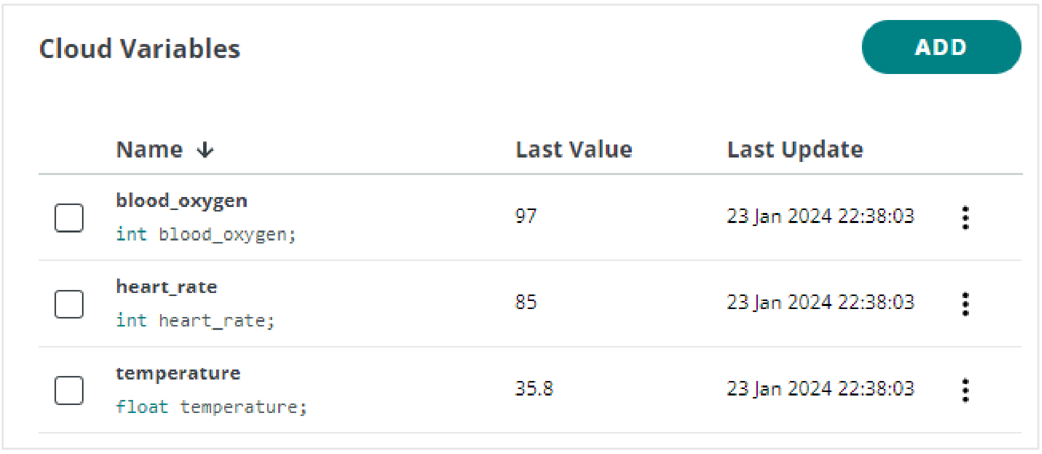
Secret Key *
••••••••••••••••••••••••••

IMPORTANT: Remember to go to the **"Sketch"** tab and upload the sketch to load the credentials on the board.

Figure 18: Wi-Fi configuration for microcontroller on Arduino Cloud

When the Arduino Cloud code is uploaded to the Arduino Nano ESP32, the Wi-Fi configuration is also transferred allowing the microcontroller to connect to the preferred Wi-Fi once the USB cable is disconnected from the laptop and it is connected to the power supply.

Arduino Cloud Variable and Dashboard Configurations



Cloud Variables

ADD

Name ↓	Last Value	Last Update
blood_oxygen `int blood_oxygen;`	97	23 Jan 2024 22:38:03
heart_rate `int heart_rate;`	85	23 Jan 2024 22:38:03
temperature `float temperature;`	35.8	23 Jan 2024 22:38:03

Figure 19: Cloud variables showing last updated sensor values on Arduino Cloud desktop version

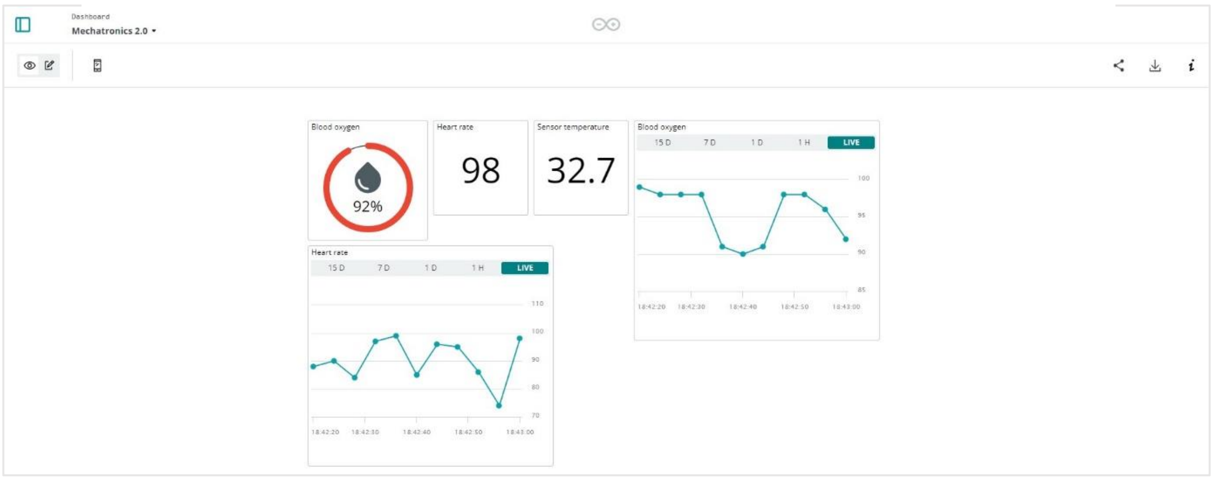
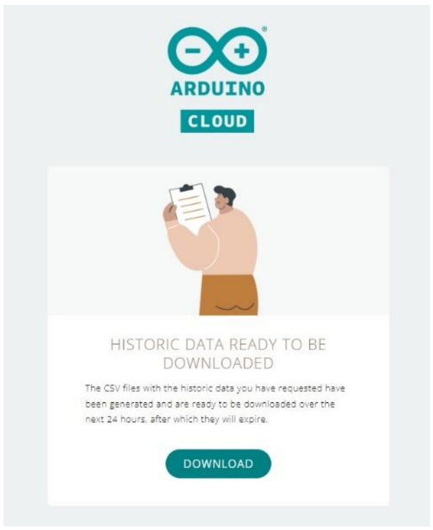


Figure 20: Arduino Cloud dashboard showing live widgets and graphs

Arduino Cloud Data Download

Historic data with timestamping can be downloaded in an excel .csv format from the dashboard and a link will be sent to users' email.

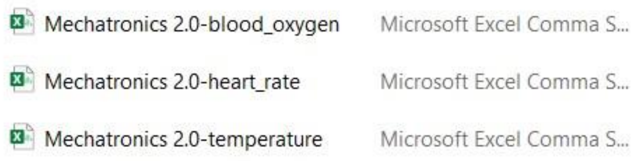


ARDUINO CLOUD

HISTORIC DATA READY TO BE DOWNLOADED

The CSV files with the historic data you have requested have been generated and are ready to be downloaded over the next 24 hours, after which they will expire.

DOWNLOAD



Mechatronics 2.0-blood_oxygen	Microsoft Excel Comma S...
Mechatronics 2.0-heart_rate	Microsoft Excel Comma S...
Mechatronics 2.0-temperature	Microsoft Excel Comma S...

Figure 21: Historic data download link sent to user email and downloaded excel .csv files

10. WORK DISTRIBUTION

Ami: Video editing and animation. Arduino IDE code, Arduino Cloud code, configuration of Arduino Cloud IoT remote app dashboard, research for components, wiring of components, and parts of the documentation write-up.

Avni: Writeup and research of components, video script and voiceover, helped in Arduino code and setup, instructions for operation use.

Adiba: Research of components, block diagram and system operations, wiring diagram, circuit diagram, designing and making the casing of the device

Aasimah: Purpose of the system and intuitive design write-up, creation of circuit diagram and data graphs, system data flow research for block diagram.

11. REFERENCES

[1] The Pi Hut. (n.d.). *Gravity: MAX30102 Heart Rate and Oximeter Sensor*. [online] Available at: https://thepihut.com/products/gravity-max30102-heart-rate-and-oximeter-sensor [Accessed 25 Jan. 2024].

[2] RobotShop UK. (n.d.). *Arduino Nano ESP32 with Headers*. [online] Available at: https://uk.robotshop.com/products/arduino-nano-esp32-with-headers [Accessed 25 Jan. 2024].

[3] The Pi Hut. (n.d.). *Mini Breadboard - White*. [online] Available at: https://thepihut.com/products/mini-breadboard-white?variant=32466763972670¤cy=GBP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gad_source=1&gclid=Cj0KCQiAqsitBhDlARIsAGMR1RhdYdwQhYzt-4UqveEJhF-Oj4ATqtgcDJ8f49LGnrMD4aSlVF-L5fYaAlVQEALw_wcB [Accessed 25 Jan. 2024].

[4] LucN5 (n.d.). *Guide to Using MAX30102 Heart Rate and Oxygen Sensor With Arduino*. [online] Instructables. Available at: https://www.instructables.com/Guide-to-Using-MAX30102-Heart-Rate-and-Oxygen-Sens/.

[5] Last Minute Engineers. (2022). *Interfacing MAX30102 Pulse Oximeter and Heart Rate Sensor with Arduino*. [online] Available at: https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/?utm_content=cmp-true.

[6] GitHub (2019). DFROBOT_BloodOxygen_S. [online] Available at: GitHub - DFRobot/DFRobot_BloodOxygen_S