Methods and materials (STARTED BUT NOT FINISHED)

This section describes the hardware configuration and software architecture of the wearable stress measurement system. The system is built around an ESP32 microcontroller and incorporates multiple physiological sensors to capture heart rate, blood oxygen saturation (SpO₂), galvanic skin response (GSR), and respiratory activity. The sensor data is processed locally and transmitted via Bluetooth Low Energy (BLE) to a mobile application for visualization and analysis.

*Arduino Nano ESP32*

The stress measuring device is powered by an Arduino Nano ESP32 which is based on the ESP32-S3 System on Chip. The main reason for using this microcontroller instead of other Arduino microcontrollers is the presence of Bluetooth and Wi-Fi. In this project, Bluetooth is used to wirelessly transmit the sensor data to a smartphone that runs software to present and visualize the stress level and the measurements. Other criteria for choosing this microcontroller include:

* Compact size
* Low power consumption
* Dual-core processor for real-time processing of sensor readings
* I2C interface, used by the MAX30102 and MAX30205 sensors for communication

Its compact size and energy efficiency make it ideal to use in a portable device.

*MAX30102*

The MAX30102 is an optical biosensor that measures the subject’s heart rate and peripheral blood oxygen saturation (SpO₂). It uses the I2C interface for communication with the microcontroller. Due to its compact design and low-power requirement, the MAX30102 sensor is optimal to use in wearable/mobile devices.

The sensor includes LEDs, photodetectors and low-noise electronics capable of ambient light rejection. Although it contains ambient light rejection, it is still advised to limit the ambient light interference to ensure measurement accuracy. The sensor uses both a red and an infrared (IR) LED to extract heart rate and SpO₂ via signal processing.

The sensor is initialized with a sampling rate of 100 Hz and is read in sequences of 100 samples to calculate the heart rate and SpO2 using an open source algorithm.

*Grove GSR Sensor*

The Galvanic Skin Response (GSR) sensor measures the skin conductance between two electrodes, which varies with sweat activity. Since skin conductance is linked to autonomic nervous system activity, it serves as a reliable indicator of stress.

It works by placing two electrodes on the skin (usually fingers) and applying a small amount of voltage. The resulting conductance is measured, with higher values indicating increased sweat gland activity. This can be the result of elevated stress levels.

The sensor outputs an analog voltage which is read using the pin A2 on the ESP32. An average over 40 samples is calculated to lessen the impact of fluctuations.

*Plux Piezo-Electric Respiration (PZT) Sensor*

The Plux PZT Sensor is a sensor that is capable of gathering basic respiration data. It includes a chest-belt with a built-in piezoelectric sensing element (piezoelectric film) that gathers data by measuring respiratory motion. The sensor detects mechanical deformation of the sensor caused by changes in thoracic or abdominal volume when breathing.

The signal is read using the analog pin A0. A peak detection algorithm is used over a 60-second window to count the inhalations made within a minute. This results in a value for breaths per minute (BPM).

(<https://www.pluxbiosignals.com/products/respiration-pzt?srsltid=AfmBOoqX-ZqraYTy9DBPU243tGuNZ-K0sgG5AEa1SrWpJkrOwiXdz8Sm>)

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Function** | **Connection** | **Pin(s) used** |
| MAX30102 | Heart rate and SpO₂ via PPG | I²C | SDA/SCL (default Wire) |
| Grove - GSR Sensor | Galvanic skin response (skin conductance) | Analog | A2 |
| Plux PZT Sensor | Respiratory effort (chest movement) | Analog | A0 |

*Table 1: Summary of sensor functions, interfaces, and microcontroller pins used*

*Software Architecture*

The firmware is written in C++ using the Arduino framework. The software handles sensor initialization, data acquisition, BLE communication, and command handling.

For BLE communication a custom BLE GATT service is implemented using the following UUIDs:

* **Service UUID**: 180C
* **Data Characteristic (UUID: 2A6E)**: Used to transmit sensor data (HR, SpO₂, GSR, and respiration).
* **Command Characteristic (UUID: 2A6F)**: Receives control commands from the mobile application.

The ESP32 is programmed to be able to support following commands:

* **START:** Begin continuous measurement from all sensors.
* **START HEART, START SPO2, START GSR, START BREATHING:** Begin measurement from a specific sensor.
* **STOP:** End all measurements and reset system flags.

Each sensor is handled by a dedicated function that enables sensor data acquisition. In each measurement loop, following steps are executed:

* **MAX30102**: Samples 100 values each from red and IR channels, processed by maxim\_heart\_rate\_and\_oxygen\_saturation().
* **GSR**: Samples the analog input 40 times with a short delay between reads and calculates the average value.
* **PZT (Breathing)**: Over a 60-second window, the analog signal is sampled at 20 Hz. A basic threshold-based peak detection method is used to count breaths.

To avoid conflicts, the sensor readings are performed sequentially, particularly for analog inputs. A delay of 100 ms is introduced in the main loop to ensure system responsiveness and allow BLE event handling.

After each data acquisition, the data from each sensor is combined into a single string (e.g., "HR:75 SpO2:97 GSR:523 Breathing:15") which is sent to the BLE client via a notification.