EpiCap–A wearable seizure monitoring device

by

Yuanrui Chen (yuanrui3)

Yichen Wu(yichenw5)

Yiyang Xu (yiyangx6)

Final Report for ECE 445, Senior Design, Spring 2022

TA: Cheng(Jamie) Xu

With help from Kenny Leung

May 3, 2022

Abstract

Our project aims to build a portable device which is able to measure patients' EEG data. There is also a small, light-weighted camera at the front of the cap to take photos together with the EEG measurements. Thus, when the data is further evaluated, physicians can pair the EEG data with the patients' facial expressions and body movements to see if there is any indication before a seizure. The EEG data and photos are stored into an on-board micro-SD-card and uploaded to the cloud, creating a potential database which is large enough for conducting research upon seizure detection using deep learning.

Table of Contents

1. Introduction

1.1. Problem

EEG, electroencephalogram, is a method to record the electrical activity of the surface layer of the brain. EEG tests are generally performed by physicians to diagnose and treat brain disorders, especially epilepsy. In order to get tested, epilepsy patients are normally required to stay overnight at the hospital so that their brain behaviors and body movements can be recorded. However, hospital bills are usually very expensive. For a single epilepsy patient, the cost ranges from 10000 to 20000 dollars per year.

The second option for those patients is the ambulatory device which is portable and can be carried home. However, it's very ugly and bulky, so those patients are less likely to carry them anywhere else. Its functionality is also very limited because it does not contain cameras, so there is no recording of the body movement available. Those patients never know when and where they are gonna experience seizures, they might have to go to the hospital during work or school, so they won't be able to have a predictable schedule. And this leads to a lower employment rate and a lower income.

OpenBCI Cyton Board is an off-shelf solution to this problem, which costs more than $700, and it does not have camera and cloud functionalities. Our product will be less costly and have additional features compared to the off-shelf product.

1.2. Solution

We wanted to create a device that can minimize its influence on the patient's normal activity. To minimize the weight of the device on the patient, the prototype should fit onto a baseball cap. Apart from the EEG measurement modules, we added a camera to the seizure detecting device. We wanted to put the camera on the front of the cap visor, recording the patient's eye and arm movements. It needs to open a stream on a specific web server that only the doctor has access to. It is also able to take quick photos at a rate of 0.2 seconds per photo when it receives a signal that the patient is experiencing seizure, and then save the photos into a local SD-card which later can be reviewed by the doctor. The EEG data can also be saved into the SD-card.

The cap can also record EEG data in real time and synchronize the data onto the cloud, a web server that only the doctor has access to. We haven't implemented the seizure detection algorithm on board, but the data collected will be extremely helpful for future research.

1.3. High Level Requirements

1. EEG data should be collected at 250 +/- 5% Hz and uploaded continuously to the SD-card and web server in real time with timestamps.
2. The EEG cap should be able to record the patient's eye and arm movement. Large number of pictures should be taken with timestamps for future analysis by doctors.
3. Every component should be mounted under the cap visor and not affect the patient's line of sight.

The final deliverable has achieved the high-level requirements.

1.4. Block Diagram

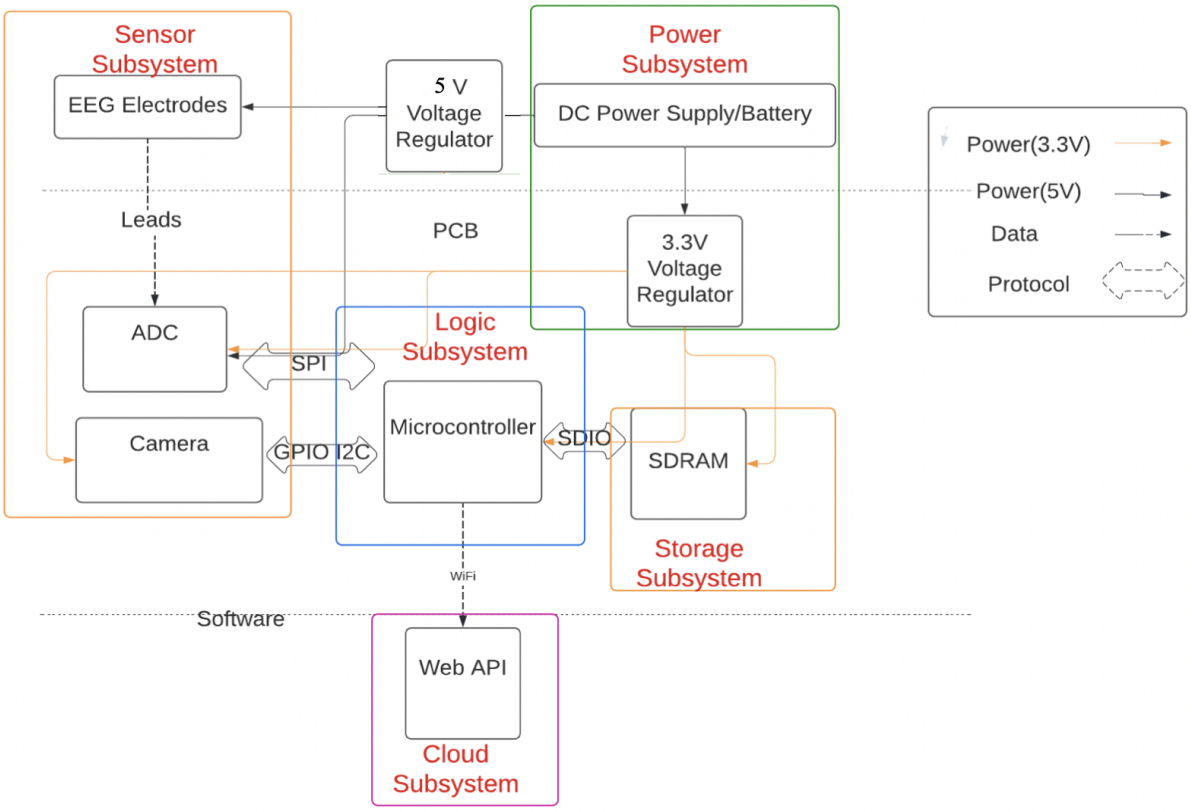


Figure 1. First Version of Block Diagram

Our final implementation is for the most part, consistent with the above block diagram. We have made five changes. First, the EEG electrodes no longer need power supply. Second, instead of using SDRAM, we use a micro SD-card for storage purposes. Third, the communication protocol between the microcontroller and the storage subsystem is SPI, since the microcontroller in use doesn't have a SDIO interface. Fourth, The cloud subsystem is based on the server provided by

the microcontroller. Web API has not yet been deployed. Fifth, instead of connecting a camera sensor to the microcontroller, we use a camera module that has its own storage, control and cloud subsystems. The reason for this is the camera interface requires many pins from the microcontroller and many of them are already occupied. Also, the camera module is extremely cheap and efficient.

We divide the project into six major blocks:

The **power** block provides steady analog and digital power supply for all the components.

The **sensor** block measures EEG data and transmits them digitally to the microcontroller.

The **storage** block stores the EEG data.

The **logic** block handles the reading and writing of the EEG data. It also processes the data and prints them to the serial plotter and monitor. Moreover, it provides a signal to the camera block so that the camera can start taking pictures.

The **cloud** block uploads the data written to the SD card to the local server. It's based on the storage block and the logic block.

The **camera** block is responsible for taking pictures when a seizure happens. It has its own storage and cloud support.

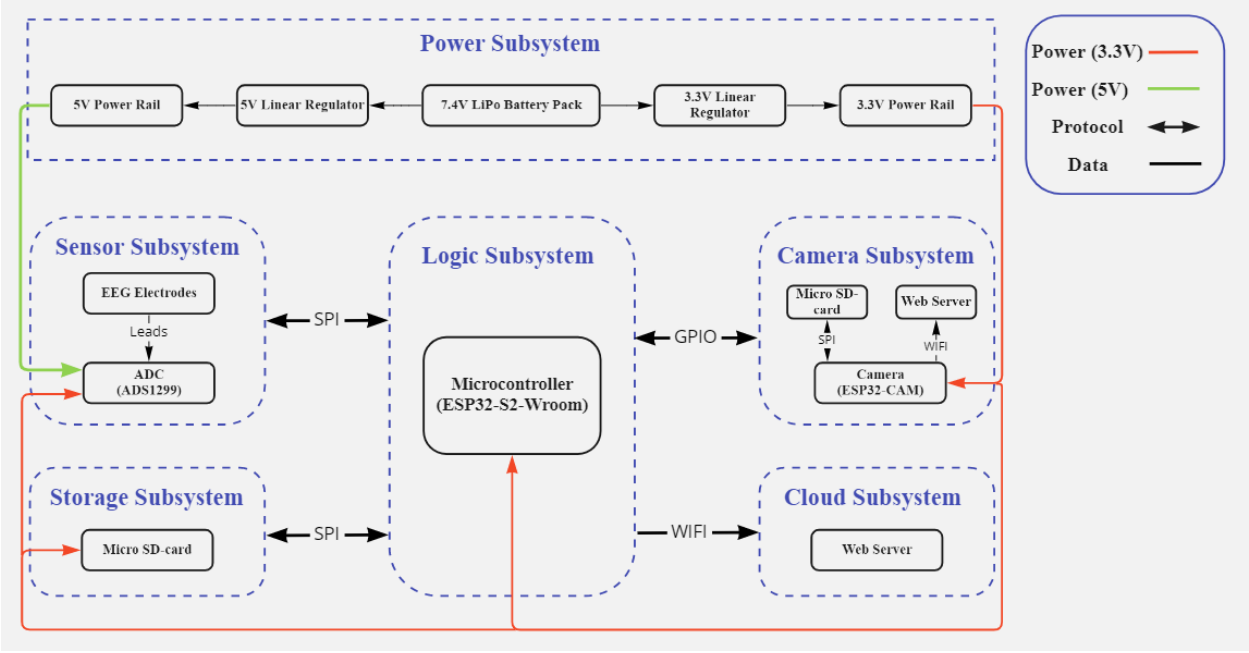


Figure 2. Final Version of Block Diagram

2. Design Procedure and Details

2.1. Physical Design

We designed a small container to hold the components like PCBs and the camera module. We created a ramp with 5 degrees at the bottom of the container in order to give the camera a better angle to capture the patient's eye and arm movements. The container is designed to be placed under the cap visor and not affect the patient's line of sight. We have two PCBs in total, one is the ADC board which includes the ADC circuit, and the other one is the main board which integrates most of the subsystems (logic, storage, power subsystems). Ideally, we will have the ADC board plugged into the mainboard to reduce the size and add flexibility. Unfortunately, the mainboard did not work out due to design errors, and we will discuss it later in the Conclusion section.

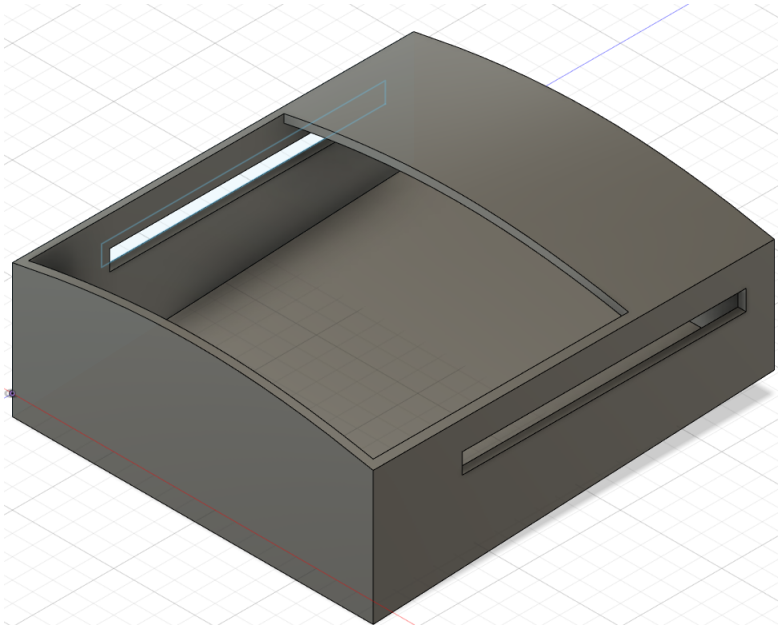


Figure 3. CAD Model of Component Container

2.2 Power Subsystem

Since safety is the priority for this project, we chose to use rechargeable LiPo batteries. Unlike lithium metal which is unstable during charging, LiPo batteries use lithium ions which have a lower energy density, thus increasing safety measures during charging and discharging[1]. It is becoming the mainstream option in the smartphone industry. The hazard of it is that it contains flammable electrolyte. So the batteries should still be handled cautiously and stay away from fire sources.

During the design process, we proposed two different schemes to provide 5V power to the circuit: use a charge pump (DC-DC boost converter) which raises the voltage from a single 3.7V battery to 5V or use two batteries piled up with a total 7.4V and a linear voltage regulator to lower the voltage to 5V. The former one occupies less space since it only needs one battery, however, charge pumps are switching regulators which usually create switching noise. It may interfere with the microcontroller's antenna which is operating in radio frequency and impact the wireless applications[2]. On the other hand, the linear voltage regulator is simpler and has lower noise on the output. Therefore, we chose the latter one and used a LiPo battery pack.

2.3 Sensor Subsystem

For electrodes, we will use the OpenBCI snap electrode cables and headset. We verified them with the OpenBCI Cyton board.

Since the project measures the EEG data, and according to Table 1, its typical frequency and amplitude lies between 0-30Hz and 2-100 μV respectively, we found ADS1299 from Texas Instrument as an appropriate design choice. ADS1299 is a biopotential ADC designed for EEG, ECG data acquisition with sampling frequency from 250 Hz to 16 kHz [3]. It also does not require any additional amplifier and isolation ICs. It has internal gain up to 24 times. It is also miniaturized.

Table 1. Frequency and Amplitude of EEG Signal

Bands	Frequency(Hz)	Amplitude(μV)
Delta	0-4	20-100
Theta	4-8	10
Alpha	8-13	2-100
Beta	13-30	5-10

ADS1299 is DC coupled and has a bias input. When measuring EEG, the bias input is tied to two ear clips. The DC bias from the human body can be subtracted from the measurements. The reference voltage is the maximum voltage the ADC can measure, and we used the internal reference voltage of the ADC, which is 4.5V.

The formula to convert 24-bit raw data to actual voltage level is given below. In our case, gain is 24 and Vref is 4.5V. The 24-bit raw data is in MSB-first format. 2's complement is used to represent both positive and negative values.

$$1\ LSB\ = \frac{\frac{2\times Vref}{Gain}}{2^{24}} =+ \frac{FS}{2^{23}} \qquad \text{(Eq 2.3.1)}$$

For the peripheral schematics of the ADC, we closely followed the OpenBCI schematics. At the input stage, a low-pass RC filter is applied to suppress radio-frequency noise leaking from clocks and antennas. The resistor value is 2.2 kΩ and the capacitor value is 1 μF. The cutoff frequency is computed below:

$$f_c = \frac{1}{2\pi \cdot R \cdot C} = \frac{1}{2\pi \cdot 2200 \cdot 10^{-7}} = 723.43 \text{ Hz} \qquad \text{(Eq 2.3.2)}$$

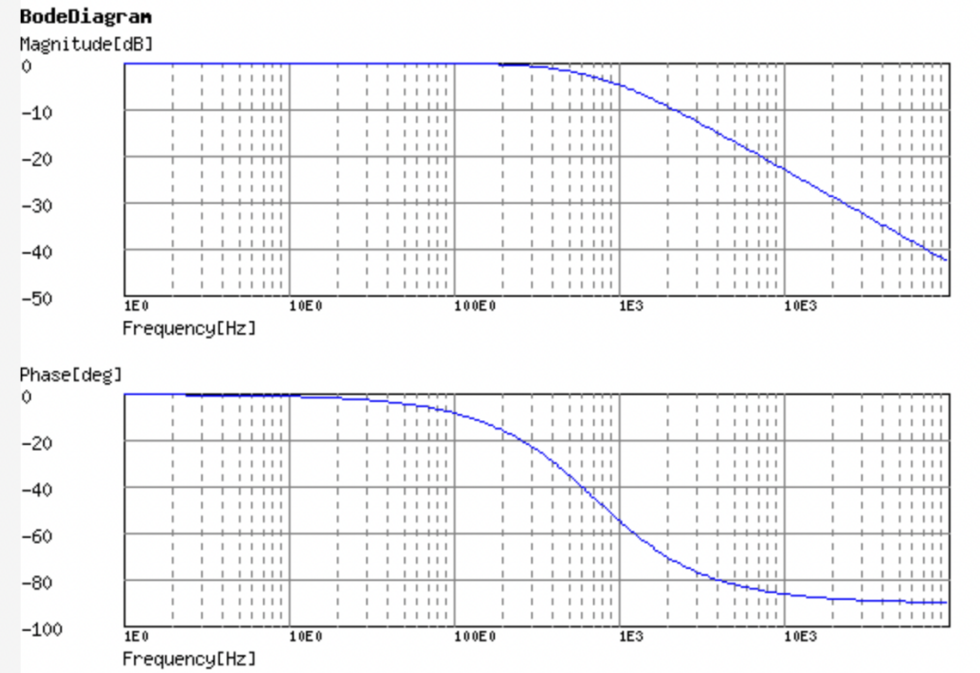


Figure 4. Bode Plot of The RC-filter

Since the ADCs are extremely sensitive to electrostatic discharge, we provide TVS diodes that provide a path of discharge. Decoupling capacitors for the reference input, power supply, and VCAP are also added to the schematics. Noticeably, we used resistor packs and diode packs in our PCB design to minimize the board area.

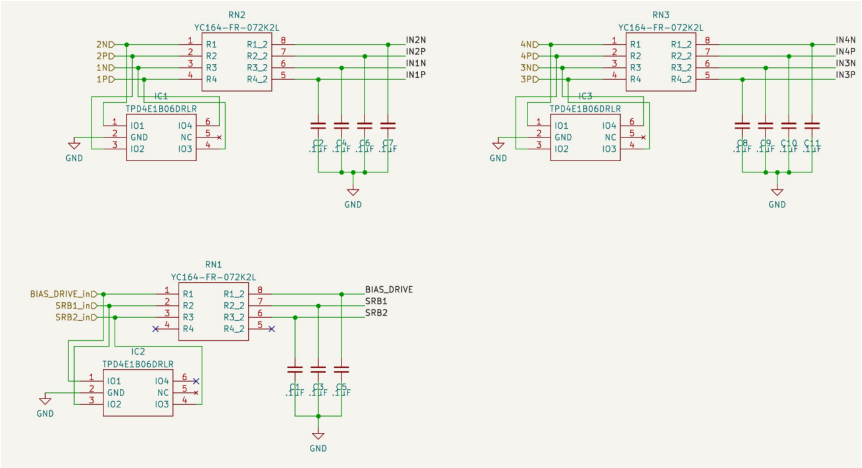


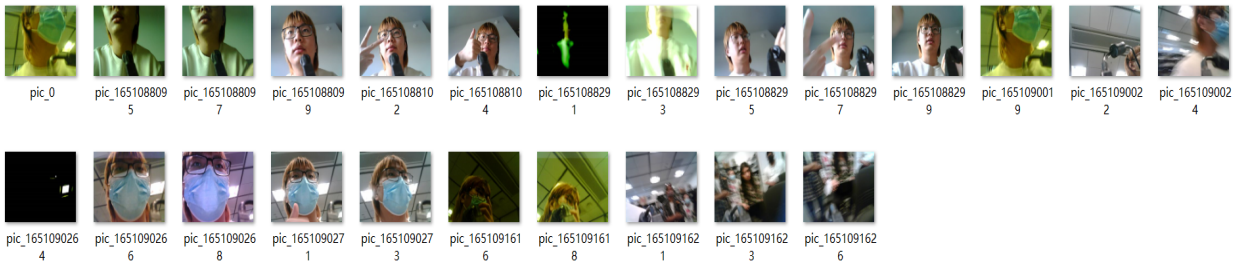
Figure 5. Schematic of RC-filter and TVS Diodes

2.4 Camera Subsystem

For the camera subsystem, we implemented a camera in front of the cap visor which is able to record the patient's eye and arm movements during a seizure event for physicians to review afterwards. To achieve this, we chose the ESP32-CAM module because it is small, and it supports WiFi and micro SD-card. We build the communication between the camera module and microcontroller by a single GPIO port, to be specific, the microcontroller will need to send a digital low signal to enable the camera module and a digital high signal to disable it.

The camera module has two modes: streaming mode and picture mode:
For streaming mode, the camera module can take videos and stream it on the web server. We used Arduino IDE to implement the WiFi onto the module through the FTDI programmer. The web server was created for the camera module by using the ESP32-CAM WiFi library, and physicians could have access by entering its unique IP address.

For picture mode, the camera can take photos at a user-defined frequency. The maximum resolution for the camera module is 1600x1200 and the fastest rate it can achieve is 0.2 second per photo. All of the photos taken will be saved locally into the micro SD-card and uploaded to the web server once connected to WiFi. Since we also implemented the NTP, network time protocol, we were able to add timestamps for each photo as its name. Along with the timestamps for EEG data, we were able to synchronize both of them by pairing up their timestamps. That allows the physicians to review the EEG data along with the corresponding physical movements in the future.



This is a photo of the picture taken and saved into the SD card with a specific timestamp.

2.5 Logic Subsystem

For the logic subsystem, we were choosing among three 32-bit MCUs, STM32-WB, ESP32 and ESP32-S2 as our microcontroller. STM32 was discarded since it's incompatible with Arduino IDE firmware. Also, STM32's wireless modules are relatively expensive. If we choose STM32, we need to assemble an additional antenna on board, which makes the project more susceptible to mistakes. Between ESP32 and ESP32-S2, we chose ESP32-S2 for its better performance. The

ESP32-S2-WROOM-I module has an on-board antenna and has a corresponding development board for us to conduct unit tests. It supports WiFi and has 2 sets of SPI, FSPI and HSPI. Since the MCU needs to communicate with both the SD-card and the ADC, 2 sets of SPIs turned out to be critical for the project to succeed.

We also designed an MCU breakout board to unit test the ESP32-S2 module. Most of the pins are broken out and we made the BOOT button and RESET button for the code to be uploaded.

2.6 Storage Subsystem

A micro SD-card was chosen due to its size and storage capacities. Since we don't need to access the memory at very high frequencies, the micro SD-card's relatively low speed is no longer an issue.

We chose ESP32-S2 module for the logic subsystem, and this module has an internal SPI that enables it to connect to a micro SD-card. When we were testing its functionality, we used a ESP32-S2 DevkitM 1 development board. It can be directly connected to an micro SD-card breakout board through GPIO 34, 35, 36, 37, with pins as CS, MOSI, SCLK and MISO. The CS pin can be moved to any other vacant GPIO. Once the development board is connected to the micro SD-card breakout board, it then can include "SD.h", "FS.h", "SPI.h" three libraries to enable the function of SPI and SD-card. Then, it can create files and folders in the micro SD-card, and write or append messages into the files created. Because we wanted to upload whatever we write into the micro SD-card onto the web server later, we created an HTML file to store all the data and messages. Since we also enabled the NTP, we can save the timestamp information with each message we appended.

For the camera module, because it has its own on board micro SD slot, we just needed to plug the micro SD card into the slot and the chip will be automatically connected. We also need to include the three libraries of the SD card: "SD.h", "FS.h", "SPI.h".

2.7 Cloud Subsystem

For the cloud subsystem, we used the ESP32-S2 module which itself supports the WiFi and bluetooth function. For testing the functionality, we used the ESP32 S2 DevkitM-1 development board and included the WiFi.h library in Arduino IDE. The SSID and password of the specific wifi are required before initializing the WiFi by using WiFi.begin(). Our function will continuously print "connecting to WiFi…" in the serial monitor if it is still looking for the specific WiFi or is trying to connect. Once it is connected, it will output a message of the IP address for this module. The web server can be accessed using its unique IP address.

For the timestamp information, we need to use the library “time.h”. This library requires the WiFi connection in order to obtain the time information from the NTP. Based on this implementation, timestamps can be added to each corresponding message and photo.

The most important part of the cloud subsystem was the real-time uploading of EEG data. In order to achieve this particular functionality, we wrote the EEG data with timestamps into HTML files and saved it into the micro SD-card. Using the HTML file enables the system to continuously update the EEG data to the web server.

3. Design Verification

3.1. Physical Design

We put every component inside the 3D-printed container and glue it to the cap. The cap is able to stay in balance. The camera sensor is able to capture the eye and arm area of the patient. The container does not affect the patient’s line of sight.

3.2. Power Subsystem

The oscilloscope measurement of the soldered board shows that the power subsystem can provide steady 3.3V and 5V outputs with ripples less than 0.1V. The load current of the enabled ADC is around 50 mV. The load current of the MCU connected with WiFi is around 150 mV. The load current of the microSD card is 30mV. Our two in-series LiPo batteries are 2000mAh. Since the voltage regulators make the calculations complicated, we can roughly estimate that the battery can sustain for around 8 hours. It’s rechargeable, so it meets the basic demand of the patients.

3.3. Sensor Subsystem

The ADC communicates with the microcontroller via SPI mode 1. We made some modifications to an online ADS1299 Arduino library. By the time the sensor subsystem was completed, the logic subsystem was still incomplete. We used an Arduino Uno board to conduct the unit test. Following the **Initial Flow at Power-Up** from the ADS1299 datasheet, we send the SDATAC command to the ADC to enter register read/write mode. We first read the ID register to ensure that the device is detected. Then we read the registers and write new values to them. We check if the writes are successful afterwards. After this, we send the RDATAC command to enter the continuous read mode. The ADC will start streaming data to DOUT pin after START is pulled high and RESET is pulled low.

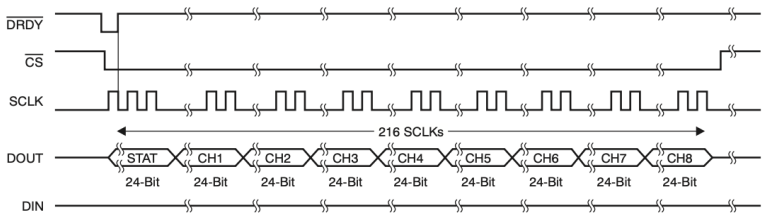


Figure 6. Expected SPI Waveform

The SPI protocol between the ADC and the microcontroller given above. The ADC operates at 2.048 MHz. The SPI rate can be set to 1.024 MHz, 2.048 MHz or 4.096 MHz. DRDY stands for data ready. Whenever the ADC is ready to send the signal, DRDY is low. The microcontroller will poll DRDY constantly to update the channel data. For pictures below, Yellow is SCLK, Green is DRDY, Blue is DOUT and Pink is CS. The first two pictures below are consistent with the expected waveforms.

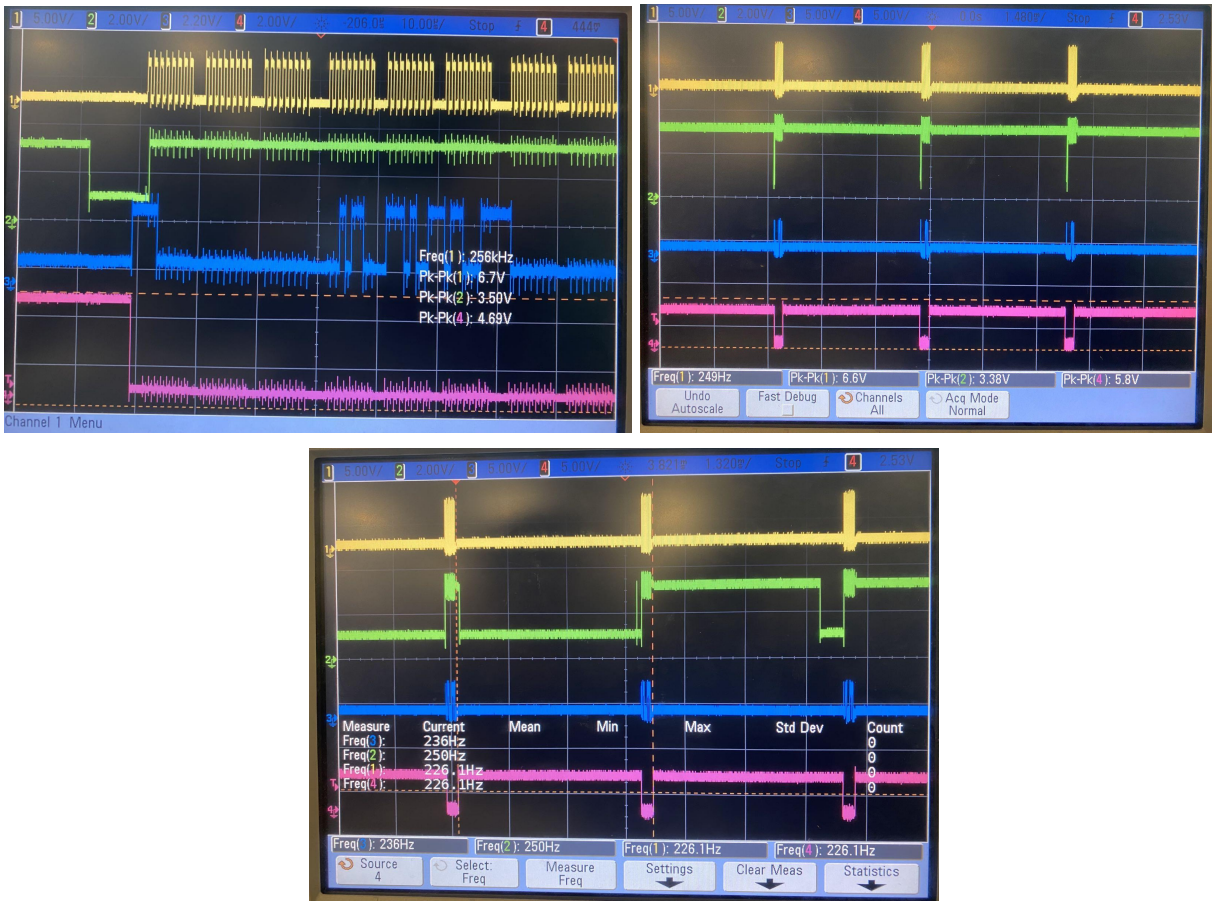


Figure 7. Experimental Results for SPI

However, DRDY may behave differently when the processor uses too many cycles for data processing between each reading. The SPI frequency will deviate from DRDY frequency, leading to cyclic misreadings. An example waveform is given below.

After verifying the SPI communication, we used the internal test signal of the ADC to verify the digital portion of the ADC. The internal test signal is a square wave and the serial plot of the data is given below. We confirmed that Pk-Pk and frequency closely match the test signal and the measurement match. The ADC is also able to measure internal noise at around 20 microvolts. With the serial plotter, we also verified that the ADC is sensitive to hand touching, as expected. However, we are yet to confirm that the EEG measurements are accurate. The reason will be explained in the uncertainties section. Also, in the serial plotter, we see overshoot reading happening every 1000 samples. This is likely due to the mismatch between SPI rate and baud rate of the serial plotter.

Figure 8. Test Results for Internal Test Signal (Square Wave from ADS1299)

We also used the serial monitor for verification. **C00000** is the status register, if it's read correctly, that indicates there is no timing issue. Also, we are able to convert the 24-bit to microvolts in real time, allowing the physicians to interpret the data. A snapshot of the reading is given below.

```
C00000, FFFC08, FFFD02, FFFCD5, FFFC6E, 1790, -9856.61, -2.39, -1.80, -1.91, -2.15
Appending to file: /ads_data.txt
Message appended
C00000, FFFC08, FFFCF0, FFFCD2, FFFC6C, 1791, -9856.61, -2.39, -1.84, -1.91, -2.15
Appending to file: /ads_data.txt
Message appended
C00000, FFFC05, FFFCFF, FFFCCF, FFFC64, 1792, -9856.61, -2.39, -1.81, -1.92, -2.17
Appending to file: /ads_data.txt
Message appended
C00000, FFFC05, FFFCF7, FFFCCF, FFFC69, 1793, -9856.61, -2.39, -1.83, -1.92, -2.16
Appending to file: /ads_data.txt
Message appended
C00000, FFFC15, FFFCFA, FFFCCC, FFFC76, 1794, -9856.61, -2.36, -1.82, -1.93, -2.13
Appending to file: /ads_data.txt
Message appended
C00000, FFFC0B, FFFD04, FFFCCD, FFFC74, 1795, -9856.61, -2.38, -1.80, -1.92, -2.13
```

Figure 9. Data Appending in Serial Monitor Interface

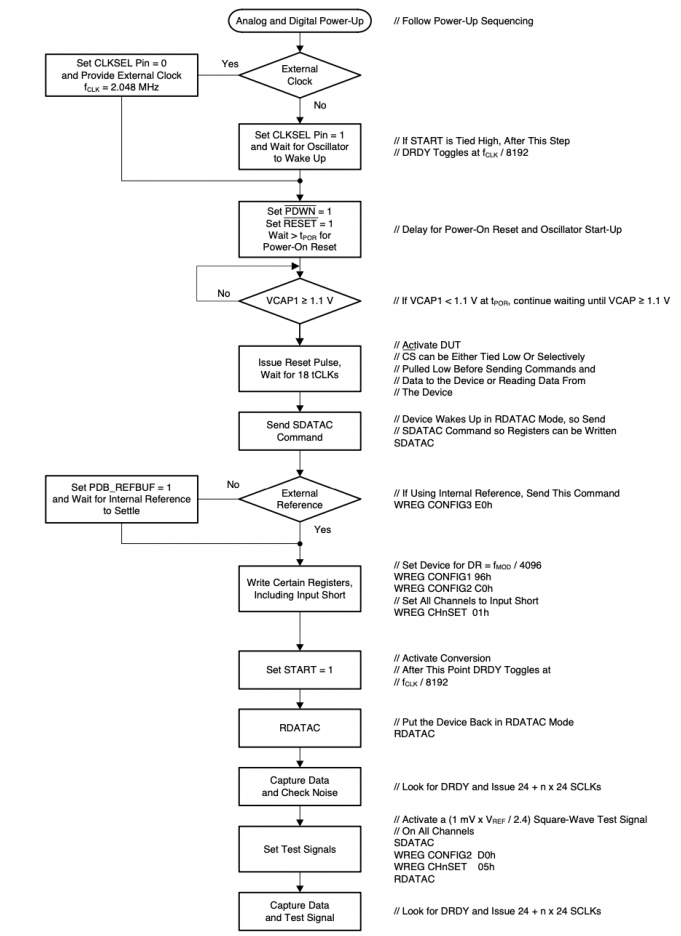


Figure 10. Initial Flow at Power-Up

3.4. Camera Subsystem

Because the video stream needs to be on the web server of the provided ip address by the camera chip so that the doctor can have access to it, we logged onto the web server using different devices, and successfully watched the correct video streaming from the camera on the correct IP address. Then we test the photos uploaded to the SD card by plugging out the micro SD card and into the computer to check, and the photos are correctly shown in the SD card each with the correct time stamp. We also changed the user-defined interval and frequency in the code, and the number and rate of the photos taken correctly changed in the way we wanted.

3.5. Logic Subsystem

The logic subsystem has many overlaps with both the sensor subsystem and the storage subsystem. ESP32-S2 is supported by Arduino IDE after we install the ESP32 board support.

ESP32-S2 has two sets of SPI interfaces. It also has two sets of SPI buses, namely HSPI and FSPI [4]. We failed to use a single SPI interface to communicate with both devices due to the hardware defect of the SD card. Thus, we assign FSPI and GPIOs 34-37 to the SD card. HSPI and GPIOs 10-13 are assigned to the ADC. The minimal viable product runs smoothly on the ESP32-S2 development board. Then we successfully uploaded the code to the ESP32-S2 module using the breakout board. The module is able to execute the same tasks as the development board does. This verifies that our project can be transferred to a PCB.

3.6. Storage Subsystem

As part of the storage subsystem is checked with the camera subsystem, we mainly focused on checking if the ESP32 S2 module correctly stores the EEG data uploaded from the ADS 1299 chip. The SD card has to be in FAT 32 bit format, which means that we could only use an SD card smaller than or equal to 32 GB. Other formats like exFAT won't be recognized by the ESP32 S2 module. Once both parts were connected, we first uploaded the code to the module and opened the serial monitor of the Arduino IDE when we were running the program. If the process succeeded, there would be "message successfully appended" printed onto the screen. Once we saw this message, we plugged out the micro SD card on the breakout board and plugged it into the computer. And we saw that each message with a specific time stamp was listed in the HTML file in the micro SD card.

SD card SPI is mode 0. The library we use assigns the SD card to the FSPI bus. Noticeably, the SD card has a defect that prohibits it to share the same SPI interface with other devices. For normal SPI slaves, when CS is high, the MISO of the slave device is high impedance. However, the MISO pin of the SD card is noisy after CS turns high, interrupting the communications of other slave devices.

The format of the EEG data is CSV. Raw data of 24 bits are sign-extended to 32 bits. They are printed in hexadecimal format. Each reading is followed by its timestamp. Every time the recording starts, a new line starting with "Begin" will be printed to indicate the header of the new measurement.

Every sample is around $5 \times 9 \times 2 = 90$ bytes. There will be $250 \times 60 \times 60 \times 24 = 21600000$ samples every 24 hours. If the ADC is open for 24 hours, 1.944 GB of data will be written.

```
FFC00000, FFFFFC11, FFFFFCFA, FFFFFCD0, FFFFFC66;
FFC00000, FFFFFC0F, FFFFFCF2, FFFFFCD7, FFFFFC5E;
FFC00000, FFFFFC11, FFFFFCFA, FFFFFCD4, FFFFFC64;
FFC00000, FFFFFC10, FFFFFCFA, FFFFFCC9, FFFFFC59;
FFC00000, FFFFFC07, FFFFFCF7, FFFFFCD2, FFFFFC68;
FFC00000, FFFFFC08, FFFFFCF3, FFFFFCD1, FFFFFC6C;
FFC00000, FFFFFC11, FFFFFCED, FFFFFCD5, FFFFFC5C;
FFC00000, FFFFFC01, FFFFFCF5, FFFFFCD5, FFFFFC61;
FFC00000, FFFFFBFE, FFFFFCFA, FFFFFCCF, FFFFFC6B;
FFC00000, FFFFFC0B, FFFFFCFF, FFFFFCD2, FFFFFC6F;
FFC00000, FFFFFC15, FFFFFCFF, FFFFFCCB, FFFFFC64;
FFC00000, FFFFFC14, FFFFFCF8, FFFFFCD8, FFFFFC5F;
FFC00000, FFFFFC11, FFFFFCF2, FFFFFCD7, FFFFFC6F;
FFC00000, FFFFFC0F, FFFFFCF7, FFFFFCD5, FFFFFC73;
FFC00000, FFFFFC0C, FFFFFCF5, FFFFFCCF, FFFFFC64;
FFC00000, FFFFFC02, FFFFFCEE, FFFFFCD5, FFFFFC5F;
FFC00000, FFFFFC03, FFFFFD04, FFFFFCCC, FFFFFC6E;
FFC00000, FFFFFC03, FFC00000, FFFFFC0D, FFFFFCFB;
FFC00000, FFFFFC11, FFFFFD02, FFFFFCCD, FFFFFC5D;
FFC00000, FFFFFC10, FFFFFCFD, FFFFFCD2, FFFFFC6B;
FFC00000, FFFFFC05, FFFFFCF9, FFFFFCD0, FFFFFC64;
FFC00000, FFFFFC02, FFFFFD05, FFFFFCD5, FFFFFC67;
FFC00000, FFFFFC0B, FFFFFCEB, FFFFFCC5, FFFFFC53;
```

Figure 11. Data written to SD card

3.7. Cloud Subsystem

The Cloud Subsystem was required to give the doctor an access to the patient's EEG data in real time. As a result, after we have successfully connected the module to the WiFi and uploaded data into the SD card, we then opened the IP address web server given by the module on another device such as our mobile phone, and we found that the data was correctly listed on the web server. And everytime we press the refresh button, there would be new EEG data shown onto the web server.

4. Cost

4.1. Labor

Labor cost estimates should use the following formula for each partner:
ideal salary (hourly rate) *actual hours spent * 2.5
We estimate our ideal salary per hour to be 40 dollar. We spend in average 20 hours per week, so the labor cost would be $40 * 20hours * 36weeks * 2.5 = $72000

4.2 Parts

Part	Supplier	Retail Price ($)	Quantity	Total Cost ($)

ESP32-S2-DevKitM-1	Mouser Electronics	9.18	1	9.18
ESP32-S2 WROOM Module	Adafruit Industry	3.25	5	15.75
ADS1299-4PAGR	Digi-Key Corporation	39.38	4	157.52
2073-MEM2075-00-140-01-ACT-ND	Digi-Key Corporation	2.06	2	4.12
3247-USDCOEM-32GB-ND	Digi-Key Corporation	9.72	1	9.72
1597-1687-ND	Digi-Key Corporation	9.97	1	9.97
ZSR500GCT-ND	Digi-Key Corporation	1.26	4	5.04
ZSR330GCT-ND	Digi-Key Corporation	1.03	5	5.15
595-TPD4E1B06DRLR	Mouser Electronics	0.58	10	5.77
963-PMK212BBJ107MG-T	Mouser Electronics	1.13	10	11.3
603-YC164-FR-072K2L	Mouser Electronics	0.07	10	0.65
80-C1206C104K3GAUTO	Mouser Electronics	2.01	4	8.04
910-SMDLTLFP	Mouser Electronics	15.84	1	15.84
910-CQ4LF	Mouser Electronics	7.95	1	7.95
80-C0805X104K1RAUTO	Mouser Electronics	0.22	13	2.8
80-C0805C105K8RAUTO	Mouser Electronics	0.20	15	3
485-1769	Mouser Electronics	0.75	6	4.5
424-PMOD-USB-UART	Mouser Electronics	9.99	1	9.99
SparkFun LiPo Charger Plus	Sparkfun Electronics	11.5	1	11.5
Lithium Ion Battery - 2Ah	Sparkfun Electronics	13.95	2	27.9
Total				

	325.69

4.3 Total cost

The total cost is $325.69+$72000=$72325.69.

5. Conclusion

5.1. Achievements

Epicap is a hard project, but we still managed to build most of the parts. We have fully verified every subsystem other than the ADC, and the digital part of the ADC is fully verified. Also we are able to run the entire system on the ESP32 S2 development board.
We have verified that the unsoldered PCB can work, and the program can successfully upload to the ESP32-S2 module with a breakout board.

5.2. Uncertainties

We have three major uncertainties. First, we have not verified that the EEG data can be accurately captured. There are three possibilities for this. The ADS1299 chip may be broken. The peripheral design for the ADC might be problematic. Or, the bias input that is supposed to come from the clip is set up in a wrong way. Another uncertainty is that our MCU+Power+SD PCB failed. The soldered board has its Vdd and GND shorted. However, the unsoldered PCB is not shorted. We are still not sure what leads to the short circuit. If the problem is solved, the project can be migrated entirely to PCBs.

5.3. Ethics

There are several concerns regarding our project. The cap confronts several risks and vulnerabilities as a result of the use of batteries, electrodes, cables, chips, storage, camera. Explosion, electric shock and mechanical danger are three dangerous factors in the design. Falling is a common scenario for epilepsy patients. We must also take precautions that our device does not create more danger for a patient in the event of a fall. We are not responsible for the mechanical part of the design, but we will try to integrate most of the electronic components on an integrated board and encase it with soft materials.

We need to ensure that both the battery and board present no hazard to the patient, especially in the event of high heat, moisture, and any sort of mechanical perturbation. Chemical leak and harmful electromagnetic radiation should be taken seriously.

When uploading the data and video to the cloud, we have to make sure that they are confidential and only the specific doctor can have access to it. The camera attached should not be deployed for any other purpose other than monitoring the patient's eye movements. The access to the camera data should be strictly obliged to medical ethics code. We can design a password system for the web API and the local storage so that the confidentiality of the information is preserved.

The design and testing of this product will comply with the IEEE ethics code. We will not use OpenBCI resources without proper citation. We will also consult doctors on the safety aspect of the final deliverable.

5.4. Future work

In the future, we want to calibrate the ADC with real EEG inputs. Before this, we have to verify that the electrodes and cables are functional. Solving the timing and overshoot issue of ADC reading is also a problem that we need to solve.

On the issue of physical design, we need to put every subsystem on one PCB and reduce the size, so that the whole system can better fit into the cap visor.

In the future we need to implement the Seizure Detection Algorithm on chip or on cloud. This is a very important one because some of the subsystems need the signal of seizure detection to activate. We also need to enable daisy chains that support more channels and develop a backend server and a GUI that visualizes the EEG data, or try to integrate with the OpenBCI Software.

References

[1] W. Walter, “Step-Down Charge Pumps Are Tiny, Efficient and Very Low Noise,” *Linear Technology.* [Online]. Available: https://www.analog.com/media/en/reference-design-documentation/design-notes/dn310f.pdf. [Accessed May 5, 2022].

[2] “Lithium-ion Polymer Battery Advantages and Disadvantages,” *Large.net.* [Online]. Available: https://www.large.net/news/8ju43nv.html. [Accessed May 5, 2022].

[3] Texas Instruments, “ADS1299-x Low-Noise, 4-, 6-, 8-Channel, 24-Bit, Analog-to-Digital Converter for EEG and Biopotential Measurements,” ADS1299 datasheet, Oct 2016 [Revised Jan. 2017]. [Accessed Feb 25, 2022].

[4] Espressif Systems, “ESP32-S2-WROOM ESP32-S2-WROOM-I” Datasheet, 2022

Appendix A: RV tables

Table 2. RV Table for Power Subsystem

Requirements	Verification	Verification Status
1. It must provide relatively low noise to the system in case of affecting the ADC (analog to digital) functionality.	1. Use an oscilloscope to measure the ripple voltage from input voltage(5V)/LDO output voltage(3.3V) to ground respectively and ensure they are less than 1 μV.	Not Verified
2. The linear voltage regulator should continuously provide 3.3V +/- 0.1V.	2. Use an oscilloscope or a digital multimeter to measure the output voltage of the LDO and ensure it's within 3% of 3.3V.	Verified
3. The battery should continuously provide 7.4V +/- 0.1V.	3. Use an oscilloscope or a digital multimeter to measure the output voltage of the battery and ensure it's within 3% of 7.4V.	Verified
4. The battery capacity should be large enough (somewhere between 5000mAh to 7500mAh) to support the circuit for at least 24 hours.	4. A. Fully charge the battery. B. Discharge it with the operating voltage. Ensure it can continuously operate for at least 24 hours.	Not verified
5. The operating temperature of the battery should be	5. Use an infrared thermometer to measure the	Not verified

within 0 to 30°C.	temperature during discharging of the battery and ensure it's less than 30°C.	

Table 3. RV Table for Sensor Subsystem

Requirements	**Verification**	**Verification Status**
1. The ADC should have a sampling frequency of 250Hz+/- 5% to ensure the quality of the EEG data.	1. A. Send stream command to ADS B. Receive the test data to the MCU. C. calculate the sampling frequency by looking at timestamps.	Verified
2. Electrodes must remain in contact with the patient's scalp during the seizure and be able to collect the EEG data continuously.	2. A. Have one of the team members wearing the EpiCap. B. Ensure the system is fully and continuously functioning. C. Make some big movements such as running and falling. D. Ensure the electrodes still remain in contact and the system can still receive data.	Not verified

3. The sampling error for the EEG data should be less than 5%.	3. A. Use a function generator to simulate the brain's electrical signals (analog signal) to one of the electrodes. B. Use Cyton + Daisy Biosensing Board to receive the signal from the electrode and record the EEG data with OpenBCI software. C. Simulate the same signal to EpiCap and compare the EEG data with the result from step B.	Verified
4. The size of the camera should be small and the weight should be less than 50g.	4. Weight the camera and ensure it’s less than 50g.	Verified
5. The camera must be able to capture the patient’s eye and arm movements during seizure	5. A. Turn on the camera and record for a few minutes. B. Adjust the angle if needed. C. Check the saved video on the SD card to see if it’s capable of capturing eye and arm	Verified

	movements.	

Table 4. RV Table for Logic Subsystem

Requirements	Verification	Verification Status
1. The microcontroller must be able to establish duplex communication with the ADC and camera.	1. a. Send “start stream” instruction to ADC via SPI. b. Observe the test data stream and check if the device is functional. c. Provide test signals to the ADC. (Depending on the number of channels we use different number of test signals) d. Probe the pins of MCU and check if the signals are transmitted correctly. e. Use an Arduino programmer to check if signals are received. f. Repeat the above steps for the camera module.	Verified
2. The microcontroller must be able to upload data given WiFi connection.	2. a. We first test the wireless functionalities on an ESP32 S2 development board. b. As we have confirmed the connection	Verified

	between MCU and ADC, we can connect the ADC to MCU. c. Provide a test signal to the ADC. d. Connect the ESP32 module with a device. e. Confirm that the test signal is transmitted without significant loss.	
3. The microcontroller should have enough RAM/cache to process EEG data and camera video data.	3. Compute the data input size and compare it against the RAM/cache given in MCU datasheet.	Verified
4. The microcontroller module must be able to determine the WiFi connectivity.	4. a. Test under WiFi coverage. b. Repeat step 2 to see if WiFi is connected. c. Probe the internal signal that monitors WiFi connection to check if it's high. d. Disconnect the WiFi and check the internal signal again to see if it's low.	Verified
5. The microcontroller is able to send signals to the camera peripheral.	5. a. Connect the corresponding pins. b. Program the MCU to provide a high input. c. Access the camera data in MCU and check if they are updated.	Verified

6. The microcontroller must write to the SD FAT32 filesystem.	6. a. Remove the content of a microSD card with a computer. b. Insert the microSD into the board. c. Partition the FAT16 file system with the firmware. d. Determine if the microSD card has been partitioned.	Verified

Table 5. RV Table for Storage Subsystem

Requirements	**Verification**	**Verification Status**
1. The storage device must be large enough to hold the video and EEG data. (Estimate 1GB).	1. a. Run the device for a period of time to check the amount of data produced. b. Connect to the pc to verify the storage space of the storage device.	Verified
2. It should be easy enough for physicians to extract the EEG/video data	2. Connect the output of the system to the pc or mobile phone and check the format of the output to see if it's easily readable by physicians.	Verified

Appendix B: Data uploaded to the server

```
1651030072
FFCO0000, FFFFFC1B, FFFFFCF9, FFFFFCD5, FFFFFC6C
1651030073
FFC00000, FFFFFC1C, FFFFFCFE, FFFFFCD3, FFFFFC6B
1651030073
FFC00000, FFFFFC0F, FFFFFDO2, FFFFFCE1, FFFFFC78
1651030073
FFC00000, FFFFFC15, FFFFFD01, FFFFFCDA, FFFFFC69
1651030074
FFC00000, FFFFFC19, FFFFFD06, FFFFFCD9. FFFFFC70
1651030074
FFC00000. FFFFFC1E, FFFFFCF7, FFFFFCDA, FFFFFC70
1651030074
FFC00000, FFFFFC1B, FFFFFD0E, FFFFFCD7, FFFFFC71
1651030075
FFC00000, FFFFFC10, FFFFFD05, FFFFFCDF, FFFFFC6C
1651030075
FFC00000, FFFFFC12, FFFFFCF3, FFFFFCDD, FFFFFC66
1651030075
FFC00000, FFFFFC1B, FFFFFCFA, FFFFFCD3, FFFFFC74
1651030076
FFC00000, FFFFFC16, FFFFFD01, FFFFFCD6, FFFFFC72
1651030076
FFC00000, FFFFFC23, FFFFFD02, FFFFFCDE, FFFFFC69
1651030076
FFC00000, FFFFFC16, FFFFFD01, FFFFFCD6, FFFFFC75
1651030077
FFC00000, FFFFFC16, FFFFFFCO, FF, FFFC1DFF
```

Figure 12. EEG Data With Timestamps

Appendix C: Github codebase

https://github.com/yuanrui3/ECE445-Final-Project