Paymodoro

Austin Lin Author, Lev Stambler Author, and David Wang Author

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— Paymodoro is a system capable of gamifying Pomodoro. Paymodoro ensures that users remain in an environment and physical state optimal for focus. If users fail to remain in this environment, they are financially penalized. If they succeed, they are paid out a pot via a lottery system. This system is unique in its combination of financial incentives and Pomodoro. We hope to increase the number of successful Pomodoro sessions a user has by at least 10% with the use of Paymodoro.**

***Index Terms*—Arduino, Blockchain, Bluetooth, ECG, Electron, Focus, Payout, Pomodoro, Sensors**

I. Introduction

Pomodoro is a technique for working efficiently. It is where you work for 25 minutes, without any distractions, relax for 5 minutes and then repeat for 2-5 cycles. Currently, there are minimal incentives and structures to ensure that a student remains focused for the duration of the 25 minutes. Temptations such as social media and messages from friends are constant; and with our decreasing attention spans, many of us simply just fail to put the effort into the 25 minute focused period. Our implementation will aim to monitor if a student is focused during their 25 minute session and penalize them if they do not remain focused and reward them if they do. We define the criteria for focus as being in a quiet environment, not moving around during the focus period, and staying in a calm state — not being excited by external stimuli.

Thus, we propose Paymodoro. Paymodoro is a system which can be used to monitor a user's Pomodoro sessions and pay them out or penalize them using Near Tokens, a form of cryptocurrency. Our algorithm determines that they remained focused for the span of the Pomodoro session. Paymodoro consists of hardware which measures and records environment sounds levels, a user's acceleration, and a user's heart rate. These measurements then sync with a desktop app which the user has open. This desktop app is also used to start and stop the Pomodoro session. Once a Pomodoro session is stopped a program running on the blockchain will reward or penalize the user depending on the results reported from the desktop app. Competing technologies include devices such as the Muse headband which use EEG to monitor focus. While their approach uses EEG, our approach uses multiple sensors as well and is directly tied to a gamified Pomodoro. Thus, while the Muse helps track focus, we directly apply focus tracking to a use case — Pomodoro.

Our primary intended user is a student looking to better their study habits and have more effective Pomodoro sessions. In particular, because we see Paymodoro as a gamified version of Pomodoro, we are targeting users who are interested in gaming.

II. Use-Case Requirements

Paymodoro must have sensors with a maximum of a 5% error range. This means that noise levels, acceleration, and heart rate cannot exceed 5% of their real value. This ensures that our measurements and subsequent calculations of whether a user is focused or not are accurate. We also require that a user is able to get a session started and ended in less than 15 seconds. In other words, from the click of the "start" or "end" button to the actual start or end of a session should take at most 15 seconds. This requirement ensures that the product is usable and does not introduce substantial overhead into a Pomodoro session.

We further require that failure criteria, sound levels, heart rate, and acceleration, (further elaborated on in the Design Section) are properly flagged in at least 95% of trials. So, if we place Paymodoro in an environment which is 10 dB more than baseline, we expect the sound criteria to be signaled as failed at least 95% of the time. The same goes for acceleration and heart rate.

We also require that the amount a user is penalized for failing a Pomodoro session to be small. This is so that a user is not scared to use Paymodoro and so that Paymodoro remains playful while still being incentivising. Thus, we require that 95% of users see the amount penalized as costing less than a cup of coffee. In the US, the average cost of coffee [1], [2] goes from $1.18 to $2.70. Thus, we require that the penalization cost be $1.

Finally, we require that there is less than a 20% false positive and less than a 10% false negative rate in accordance with our observations of users. This means that if we have a user trial Paymodoro intentionally remains focused, Paymodoro should have an error rate of less than

10%. If a user tests Paymodoro, intentionally doing unfocused activities, there should be an error rate of less than 20%. A smaller false negative rate is important as a false negative causes a financial penalization and is thus worse to have than a false positive.

III. Architecture and/or Principle of Operation

Our system consists of 3 main seperate systems:

1. The Arduino and sensors
2. The Desktop App
3. The Blockchain SmartContract

There were no changes from our design report to the system architecture. The block diagram relating all the components can be found in Fig. 1. The accelerometer and optical heart rate sensor interface directly with the Arduino. The Arduino then interfaces via Bluetooth with the desktop app. The desktop app then collects microphone data from the computer's onboard microphone and interfaces with the blockchain smart contract.

The Arduino, accelerometer, heart rate sensor, and Bluetooth chip will be housed in a battery powered box that can be clipped to the user's belt, hereafter referred to as the Belt Clipped Device. The wiring diagram is shown in Fig. 3 and the final prototype of the Belt Clipped Device is shown in Fig. 4. The Belt Clipped Device will have wires coming out of it for the optical heart rate sensor since it needs to be attached directly to the user's ear or finger.

The desktop application was written with React node.js with an electron extension to make it desktop app compatible. Furthermore, we included the Bluetooth communication to receive the data from the Arduino every 2 seconds which was written to a CSV file due to different node version dependencies with the node-Bluetooth library. The desktop application then reads from the CSV file and the data from the computer's built-in microphone to run the algorithm to determine the focus state. Once a session is completed, the desktop application makes a function call to the blockchain to notify the blockchain with the updated status of the focus session, adding or deducting Near tokens to the user's account.

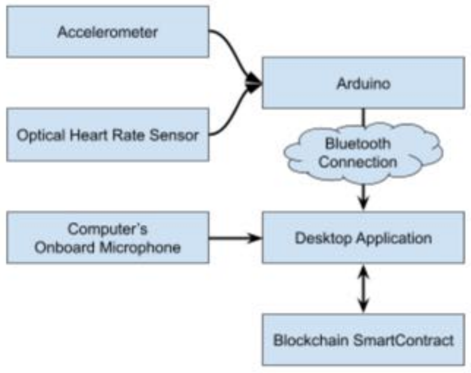


Fig. 1. Our block diagram of different subsystems

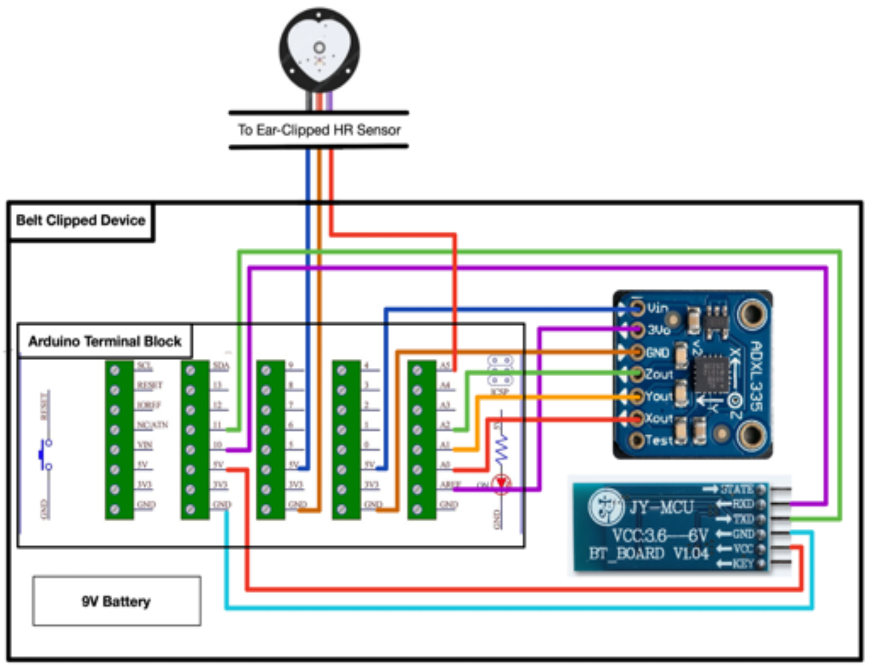


Fig. 2. Wiring diagram for the components inside the Belt Clipped Device. Inside the belt-clipped device there will be an accelerometer, a Bluetooth module to communicate with the desktop app, a battery to power the device, and wires leading out of the box to an optical heart rate sensor.

IV. Design Requirements

A. *Algorithm Requirements*

A baseline measurement will be taken at the beginning of each session of the user's environment sound level in dB and heart rate in beats per minute. Every second, a sample will be taken from the microphone, accelerometer, and heart rate sensor

For the purposes of determining whether a user has failed a session, we define two types of failure:

1. Session failure
2. Criteria Failure (Heart rate, Acceleration, Sound)

1) *Session Failure*

Session failure means that the user has failed the session and will not have the lock amount returned to them at the end of the session. There are 2 ways to fail a session. The first way to reach a session failure is having 2 or more criteria failures (defined in Section IV.A.2) over an entire contiguous 30 second period. The second way to fail a session is if one criterion fails for more than 25% of the Pomodoro session or 6.25 minutes.

2) *Criteria Failures*

Microphone Criteria Failure:

If the noise level exceeds the calibration noise level by 5 dB for every sample during a 15 second time interval, we consider this a failure. We choose 10 dB initially as 10 dB relates to the difference in noise level of a normal conversation, 60 dB, and a vacuum running, 70dB. Thus, we considered a 10dB noise increase to be "focus breaking" as the turning on of a vacuum, hair dryer, and other sounds around 70dB are enough to break concentration if the initial environment just contained normal conversation. But, we changed to a 5 dB increase based on test results.

Accelerometer Criteria Failure:

Since our accelerometer data is triaxial and includes acceleration due to gravity, we calculate acceleration by taking the magnitude of the pairwise difference between two acceleration data points. This is a change from our design report where we aggregated the acceleration data by taking the average magnitude of samples in a 15 second interval.

According to Researchgate, the acceleration a person experiences during the first 5 seconds of walking is 0.2g. Thus, if a person is experiencing 0.2g for each sample during a 5 second interval, we assume that they are starting to walk. We consider moving around and changing location as a focus breaking activity. If the difference in acceleration exceeds 0.2g for more than 5 seconds, we consider this a acceleration criteria failure.

Heart Rate Sensor Criteria Failure:

If the average heart rate is more than 40% above the calibration heart rate for the 2 second period, we consider this a criteria failure. According to a University of Oregon Study, a 40% increase in heart rate corresponds to going from a calm state to an excited state. Thus, we correlate a change from calm to excited with a break in focus. Some of these measurements have since been updated and will be mentioned in the later sections.

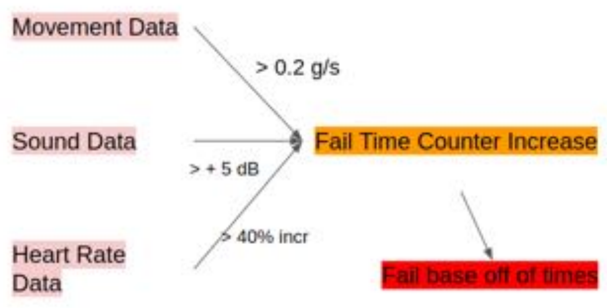


Fig. 3. A basic flow chart of the algorithm's component's

B. *Smart Contract Requirements*

In order to meet the latency requirement of each action taking at most 15 seconds, the blockchain must be fast. We specifically want a latency of less than 5 seconds from the time of submitting a transaction to the blockchain to the time it is confirmed. This will leave us 10 seconds of slack latency for dealing with the device, desktop app, and computation. Also, we require that the smart contract be inexpensive in order to keep to the use case requirement that a user can lose only around $1 dollar per Pomodoro Session.

Blockchains have fees, often referred to as gas, which pay for the computation done on the blockchain. We thus require that the amount of gas spent for initializing and ending a Pomodoro session sums to at most 10 cents. This allows for the rest of the dollar (90 cents) to go into the payout pot and thus keep the Pomodoro sessions incentivized.

C. Electron App Requirements

In order to properly connect the blockchain to the hardware, we need a desktop application to serve as a control. The blockchain needs to know a focus session has succeeded or failed, and the hardware needs to know when a focus session has begun or ended. Furthermore, we also need to be able to process the signals being sent out of the hardware. This will be done via a desktop application using the Electron App library.

The Electron App will send and receive start and stop signals to the Arduino via “node-Bluetooth” which is a Bluetooth serial port communication package for Node.js. After establishing connection with the Arduino device, the desktop application will be able to send start and end session commands to the device.

Once the desktop application receives the signals back from the Arduino device and onboard microphone, it will process the signals via a program written in Javascript that calculates the focus using the algorithm we developed. This temporary result will then be kept in local memory; and when the session ends, it will send the final results to the blockchain via the smart contract. Furthermore, the signals will also be displayed visually to the users via our front end UI to provide users with real time feedback on whether they are focused or not.

Finally, the desktop application will also fetch the resulting reward from the smart contract and display the amount of tokens earned to the user.

D. Hardware Requirements

To effectively monitor whether the user is in an environment conducive to focus, we will collect the user’s heart rate (from an optical HR sensor), acceleration (from accelerometer), and surrounding environment volume (from the computer's onboard microphone). These measurements should be accurate to within ±5% of the actual value.

The Arduino will collect measurements from the heart rate sensor and from the accelerometer every second. Every two seconds, the Arduino will take the average heart rate over the last two seconds from the heart rate data and send the magnitude of the vector difference between the two triaxial acceleration measurements. These two pieces of data will be packaged into a bit stream to be sent serially to the desktop app over Bluetooth as the Bluetooth module we are using requires a serial bit stream. There should be no more than a 0.25 second latency between the end of each 2 second interval and the time that Bluetooth transmission starts.

To detect changes in the environment sound level, the microphone onboard the computer will directly interface with the desktop application. The sound in a library is around 40 decibels. Therefore, the microphone must be sensitive enough to capture sounds above 40 decibels. Processing of the microphone input will be performed by the desktop application directly. The requirements for processing the microphone input is described in Section IV-A.2.

V. Design Trade Studies

A. Design Specification for Algorithm

When considering the algorithm, our main focus was to determine which types of body metrics to take into account. After significant research, we determined that heart rate (ECG), sound (microphone), and movement (accelerometer) were key indicators of whether a person is focused or not. Another factor we considered was potentially EEG which is brain waves. However, that could be very difficult to calibrate as the user would have to wear a headset and we would need to differentiate what types of brain activity is part of the focus and what activity is the user doing something not task related. Another consideration we had was to consider eye movement to see if the user is focusing on the correct part of the screen. However, this could violate some privacy issues as we would require the user to allow us access to their computer screens, and it would not be beneficial if the user is working on something that cannot be captured by our webcam. Our algorithm works by assigning failure or success to each individual measurement, which we call criteria, per two seconds. If a majority (⅔+) of criteria fail for a period of 30 seconds or more, then we consider the overall session to be unsuccessful. We chose a majority of failing criteria to signify a break in focus because it is possible that external events may affect the individual criteria. For example, a dog could start barking and the microphone criteria could fail, the user may get excited after solving a problem and their heart rate spikes, or the user has to move due to the library closing. By requiring a majority of criteria to fail, we decrease the probability of signaling that a user has failed when, in reality, uncontrollable events caused the user to fail.

B. Design Specification for Smart Contracts

When considering what blockchain we were going to use, we had to choose a fast and inexpensive blockchain. We considered three widely used smart contract blockchains used today, Ethereum, Solana, and Near. Fees on Ethereum, for the past year, have always exceeded $1 per smart contract transaction, so we cannot use Ethereum. Solana has had fees around 1 cent per transaction, but they have recently been having problems with transactions not confirming. Thus, some transactions can be confirmed within a second, but the same need to be retried multiple times. Thus, the latency of using the smart contract can exceed 10 seconds. Near Protocol meets our smart contract requirements. Its latency time is around 3 seconds, with fees per transaction totalling under 1 cent. Thus, if there are two transactions, one for starting a session and one for ending, the fees total to 2 cents.

C. Design Specification for Desktop App

When considering which framework to use for our Desktop Application, we looked at both QT and Electron App. We realized that QT was well suited for a richer UI and custom components and design. However, the focus of this project was not to create a rich user interface and a good looking desktop application, but rather a robust algorithm and hardware design. Therefore, we decided to go with Electron App which is simple since we can utilize CSS/HTML and Javascript. Furthermore, Electron App is also compatible with web developments which also allows us to potentially be more versatile in the future if we are considering also opening up possibilities of web applications.

D. Design Specification for Hardware

We considered 3 different options for acquiring data from the sensors. Our first design had all sensors attached to an Arduino. The Arduino would connect to a Raspberry Pi which will then communicate with the desktop application. We decided against using both an Arduino and a Raspberry Pi since both devices have general purpose IO pins and 2 devices would increase the complexity of our system which could lead to more setbacks and delays.

For our second design, we decided to connect our sensors to either an Arduino or a Raspberry Pi to transmit data to the desktop application. We decided on using an Arduino since we do not need a graphical user interface for our Belt Clipped Device.

Finally, in our initial designs, we had planned on purchasing a microphone and building an amplifier circuit which would connect to the Arduino along with the other sensors. However, since our system already specifies a computer to run the desktop application and most computers come with an onboard microphone, we decided to use the computer's onboard microphone and interface it directly with the desktop application.

VI. System Implementation

A. Algorithm

The algorithm will be implemented in TypeScript and run locally on a user's laptop within the desktop app. The desktop app will collect the sensor data via the Bluetooth connection to the Arduino. Originally, we were planning to collect data every 2 seconds from the Arduino. We changed this to every 1 second in order to have faster visual feedback from the desktop app. The app also changed to store the sensor measurements for each period in a CSV file rather than in memory. Then, after each data update, the desktop app will also calculate whether any of the criteria failed and store the failure periods. At the end of the Pomodoro session, the desktop app will calculate if the user failed 2 or more criteria at a time for 2 or more consecutive 15 second periods (30 total seconds) or if the user failed 25% (6.25 minutes out of 25 minutes) or more periods for any 1 criteria.

B. Smart Contract

We have defined the following state which must be kept

- Contract Confiscated Balance: the amount of Near which the contract confiscated from unsuccessful users before a payout is made to a successful user. A balance is an unsigned 128 bit integer type according to Near's specifications.
- Active Users: a list of user IDs currently engaged in a Pomodoro session. On Near, each user has an ID which corresponds to a string. Only users who own the ID can add or remove themselves to the list. A caller's user ID can be fetched from the smart contract with the env.predecessor_account_id() function in Near's runtime
- Lock Amount: the amount of Near required to be locked into the contract before starting a Pomodoro

Session. We are choosing ~0.1 Near for the lock amount as specified in the design requirements. This also has an unsigned 128 bit integer type
- User success rate: the number of times a user was successful and the number of overall sessions. This state was not included in our initial considerations but is required.

The smart contract also requires the following methods to interact with the state:

- Start Pomodoro Session: starts a Pomodoro session and adds the user to the **Active User** list. Calling this function requires attaching **Lock Amount** of Near. This means that the user simultaneously calls this method and transfers ~0.1 Near to the smart contract. If the user is already in the **Active Users** list, this method will fail and refund the user.
- End Session: End a Pomodoro session. If the caller signals a success, then they will get rewarded all earnings in the **contract confiscated balance**. This means that the balance will be transferred to the user. The **lock amount** will also be transferred back to the user. If the user is unsuccessful, then c**ontract confiscated balance** increases by the lock amount.
- Session results: Get the number of successful and total number of results. This method was added after our initial considerations to improve the desktop UI/ UX.

C. *Desktop App*

The Desktop App's main purpose is to determine criteria and session failure based on the specifications in IV.A and to serve as a communication platform between the blockchain and the hardware. Its secondary purpose is to serve as a user interface for the users to provide feedback.

To accomplish its primary focus, it is crucial that the Desktop app can communicate clearly with the hardware and blockchain. The hardware communication is completed via Bluetooth, and through any type of information transmission, there are possibilities that the data being transferred could be damaged. Therefore, it is important that we take measures to ensure the accuracy of the data being transmitted.

To accomplish our secondary focus, we utilized an iterative design process to verify what the user needs to see on the user interface. We first created lo-fi prototypes and conducted user testing to see what type of information is the most important part to the user, and iterated on the feedback received.

D. *Hardware*

In order to effectively monitor whether the user is in an environment conducive to focus, we will collect the user's heart rate, acceleration, and surrounding environment volume. The user's heart rate will be determined via a optical heart rate sensor. The user's acceleration will be monitored by an accelerometer in the Belt Clipped Device. A picture of our hardware system is shown in Fig. 4. Finally, the environment sound level will be detected by the computer's onboard microphone.

The Arduino will continuously collect measurements from the heart rate sensor sensor and from the accelerometer directly, aggregate these measurements as described in section IV.D, and send them serially to the desktop app over Bluetooth.

To detect changes in the environment sound level, the microphone onboard the computer will directly interface with the desktop application. Thus, the specification for how the microphone's signals will be processed is discussed in Section VI-A.2.

VII. Test, Verification and Validation

A. *Results for Design Specification of Algorithm and Hardware*

We tested the algorithm by trying to explicitly fail each criteria, recording false positive and negative rates, and seeing how our specified parameters affect the overall performance. We tested sound level increase by putting the device in a quiet environment and then speaking next to it. We found that the initial 10 dB increase threshold was too high. Thus, we changed the increase threshold to 5 dB. After running 20 subsequent tests, we had a 95% success rate where 19 trials triggered a failure of sound.

We then tested movement by sitting down and standing with the device. We found that we had to update our algorithm to look at the distance between accelerations of two sequential timesteps rather than the current acceleration value. This modification allowed us to account for gravity. After this modification and changing the threshold to 0.2 g/s, we had 18 successful trials and 2 failed trials giving us a 90% success rate.

Next, we tested the Heart Rate monitor. Unfortunately, we found that the heart rate readouts were inconsistent which caused difficulties in algorithm testing. When measuring heart rate after jumping jacks though, we had a success rate of 30% where the algorithm signaled a failure 6 out of 20 times.

Finally, whenever we tested a failure of a single criteria for more than a quarter of the time, the algorithm indicated a failure result for the session. Whenever we had a continuous failure of two criteria for more than 15 seconds, the algorithm also indicated a failure as intended.

B. *Results for Design Specification of Blockchain*

The blockchain program worked as intended. Over 20 recorded tests, we found that blockchain communication for starting and stopping a session always took less than 10 seconds. We also found that initially signing into the blockchain took less than 10 seconds 90% of the time, but slightly exceeded 10 seconds (at around 15 seconds) 10% of the time. We also found that blockchain fees never exceeded 1 cent as intended.

Moreover, the blockchain never had any consistency problems or hangups. In other words, every transaction submitted to the blockchain was successfully recorded without need to resend the transaction.

C. *Results for Design Specification of Desktop App*

The desktop application worked as intended. We conducted over 20 recorded tests for the start/end latency of the application and found that we had a result of <8 seconds to start and stop a session. Furthermore, we were able to redirect to the blockchain pages for login, and also connect successfully to the device via Bluetooth.

Still, we found that Bluetooth connection took 20 seconds on average. Though we did not specify a bound on Bluetooth connection as part of our design requirements, we found connection time to be rather slow as compared to the rest of the flow.

D. *Validation*

After running 20 trials of shortened Pomodoro sessions (lasting 1 minute), we found we had a 10% false positive rate and a 10% false negative rate. So, we had 2 false positives and 2 false negatives. This matches with our use case requirement of less than or equal to 20% false positives and less than or equal to 10% false negatives. We believe that we could have further decreased the false positives if we had a more finely tuned heart rate sensor. As for the false negatives, one of them could have been averted if our heart rate sensor did not give faulty results.

VIII. Project Management

A. *Schedule*

Our final schedule is shown in Fig 2. In the design report, we did not allocate time for system testing and designing and building the physical prototype of the Belt Clipped Device. The last 3 weeks were dedicated to testing and building the Belt Clipped Device. Work on the desktop application commenced 1 week later than originally planned because we realized that more extensive research on react, electron, typescript, and node-Bluetooth communication was needed inorder to start writing code. These are the only changes to our schedule from the design report aside from adding more detail about what was actually completed each week.

B. *Team Member Responsibilities*

Austin worked on writing the Arduino code for data acquisition from the heart rate sensor and accelerometer and also the code for Bluetooth communication with the desktop application. Austin also designed and built the prototype of the Belt Clipped Device. David worked on implementing the desktop application. Lev worked on the smart contracts. The entire team worked on integrating the systems together. We also stepped in and helped each other out when there were issues with implementation.

C. *Bill of Materials and Budget*

The bill of materials is located in Table 1 on the last page of this document. Items in red are materials that were purchased but not used. Items in yellow are materials that were not in our design report. Most of the items in yellow are materials used for building the Belt Clipped Device which was not accounted for in the design report.

D. *Risk Management*

One of the risks that we identified initially was that we could fall behind schedule due to implementation setbacks. However, this was actually not the case. In terms of implementation we were mainly on schedule, but we did have issues with parts being ordered. For example, our Bluetooth device got burnt out as soon as we first connected it to the circuit, and thus we weren't able to continue testing the hardware device until we were able to get a replacement part. We were able to mitigate this issue by ordering another part and also a spare so that if it burns out again, we would not have another time delay. Furthermore, we also switched out the tasks and spent our time working on the front end application which was scheduled to be completed the following week. Being agile and flexible in our scheduling allowed us to get back on schedule when unforeseeable issues happen.

Another risk that we had was with the ECG device. The ECG device came from Ukraine, and when the war broke out, our order was delayed. Because we kept a close eye on the tracking information, we were able to identify the issue immediately and order a different heart rate sensor. However, with the new heart rate sensor, we had issues with the accuracy of the readings we were receiving. The heart rate sensor uses optical methods to read the heart rate, which is very different from the ECG device. This also resulted in a limitation as to where the device could be placed. Thus, we had to change the device to be connected to our ear instead of the wrist. The ear also resulted in another problem of having a stable connection, thus we had to mitigate this by lowering the threshold such that there would be less false negative results.

IX. Ethical Issues

Currently, most of the projects out there are simply timers to help the users track how long the session has lasted. With our project, because there is an incentivization aspect to the project, it could potentially be taken advantage of by malicious users. Those malicious users may want to simply make money with this platform, and thus they could not utilize the device properly. For example, they could simply strap on the device and go to sleep. If they don't move a lot in their sleep and do not snore, it may be possible the device may recognize the user as being focused. This would then be unfair to the users that are working hard and actually focusing. Potential ways to mitigate this is to add in a camera that tracks users' eye movements and make sure they are in front of the computer. However, there are also downsides to this approach because it could be considered invasive to users' privacy and thus deter many users from using our device.

X. Related Work

We looked at various different types of applications that supported the Pomodoro technique and found that most of the applications were timer related.

1. Pomodoro for a simple web-based Pomodoro timer: This timer allowed users to simply access a web-based application and click start and pause times for a specific start of a focus session. They also allowed for timers for short and long break periods.
2. Marinara Timer for a shareable web-based Pomodoro time: This timer is also a web-based timer that allows users to use a timer to monitor their Pomodoro sessions. However, the one twist is that users have a unique link that they receive to share with other people because research has shown that the Pomodoro technique is even more effective when used in a small group setting.
3. Forest for a mobile Pomodoro timer: This mobile timer uses a very simple technique to disallow users from being distracted by their phones. This phone app starts by having a user plant a tree; however, if you navigate out of the phone app, the tree will die. Therefore, it encourages the user to not use their phones and focus on their current task at hand.
4. Be Focused for Apple users: This is a menu bar application that can also be linked with your iPad and other iOS devices. However, on top of the timer itself, it allows the users to jot down to-do-list items which allows them to have a better idea of what they are going to complete within this Pomodoro session.

After our related-work studies, we realized that most of these applications out there offer mainly a timer for the Pomodoro sessions – sometimes with a small twist. However, none of them have a hardware component that actually monitors how focused you are. Furthermore, they do not provide any monetary incentives for the user.

XI. Summary

The majority of our system was able to meet our design specifications. The only part of our system that did not meet our design specifications was the heart rate monitor section. This bottleneck was mainly due to the fact that our originally ordered ECG device was stuck in Ukraine due to the current war. Thus, we had to order a different device, which uses optics to determine the heart rate. This required the device to be attached precisely to the skin, and had many limitations as to which parts of the body it worked on. For example, the wrist was no longer an option because the device was unable to track the heart rate there. A simple fix we could have done was to order another ECG device earlier so that it would arrive in time for our integration.

A. Future Work

In the future, we would plan on removing the optical heart rate sensor and replacing it with an uECG sensor as originally planned. This would allow for more accurate and consistent readings. Moreover, we would add eye tracking through computer vision as a means of checking whether the user is engaged with their task. This would allow for more fine tuned determination of whether the user is focused. Finally, we would decrease the size of the containment box to make it easier to attach to the body.

B. Lessons Learned

For students in other groups, we suggest they definitely be very cautious when ordering parts and make sure that there are enough parts for you to use. Not only did we have issues with the ECG device, we also had an issue with the Bluetooth device burning out. Thus, for parts that are difficult to obtain, if it is within the budget, try to order a backup part, because if parts do not arrive or burn out, it could significantly take you off your schedule.

References

[1] *Americans Pay an Average $2.70 for Coffee, While Tipping ...* https://www.usnews.com/news/blogs/data-mine/2015/09/29/americans-pay-an-average-270-for-coffee-while-tipping-20-percent.

[2] Sherman, Elisabeth. “New Study Found the Most, and Least, Expensive States to Buy a Cup of Coffee In.” *Matador Network*, 9 Dec. 2021, https://matadornetwork.com/read/coffee-cost-state/.

-

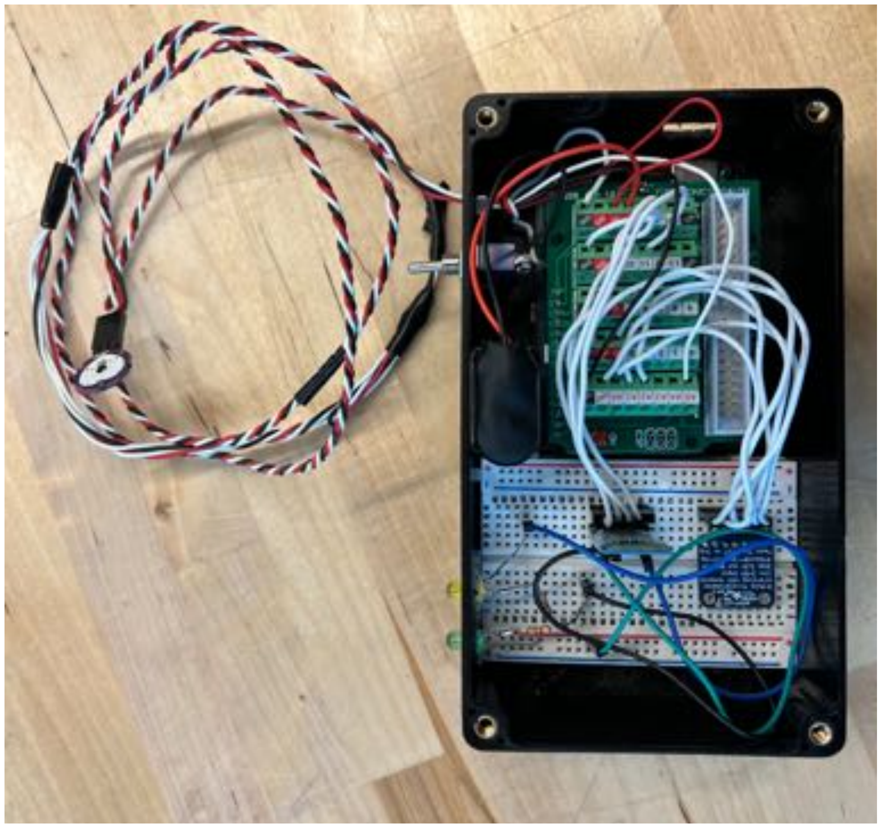
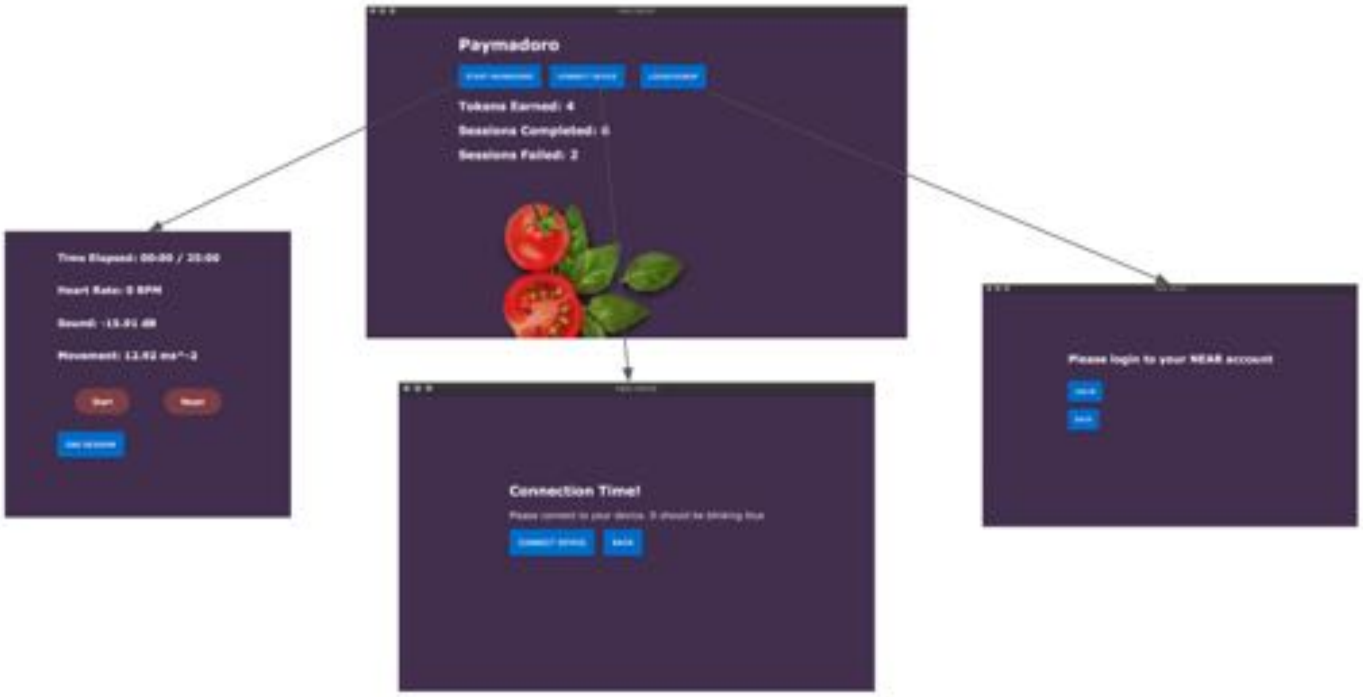


Fig. 4. Final prototype of the Belt Clipped Device. The box contains two status lights to determine if the box is powered (green) and if there is a Bluetooth connection (yellow)



Paymadoro

Tokens Earned: 4

Sessions Completed: 6

Sessions Failed: 2

Time Elapsed: 00:00 / 25:00

Heart Rate: 0 BPM

Sound: -15.01 dB

Movement: 11.82 ms^-2

Connection Time!

Please connect to your device. It should be blinking blue

Please login to your NEAR account

Fig. 5. Our desktop application with flow of which page every button leads to.

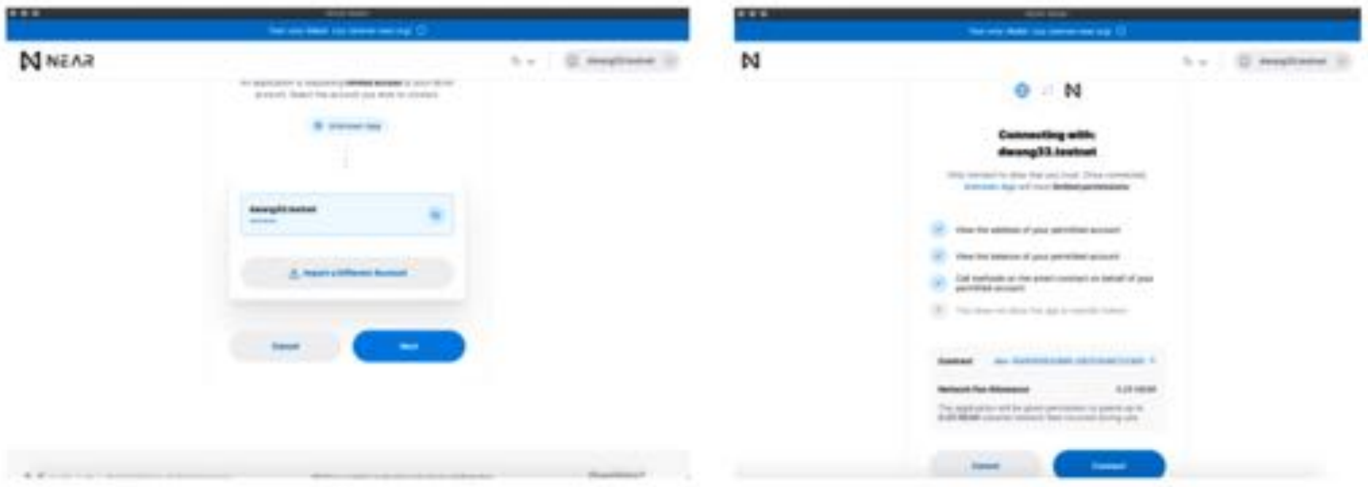


Fig. 6. Redirect pages from the desktop application to the Near blockchain for login information

Description	Model #	Manufacturer	Quantity	Cost	Total
Accelerometer	ADXL335	SparkFun Electronics	2	14.95	29.9
Bluetooth Module	8541554474	HiLetGo	6	8.59	51.54
Arduino Uno Rev3	A000066	Arduino	1	-	-
Heart Rate Sensor	B09NQ9WGCY	BANRIA	3	8.12	24.36
Molex Connector and Crimper	A7912F-FBA	A ABIGAIL	1	26.99	26.99
Arduino Terminal Block Shield	B08LH5TCM5	Xiken Electronic Technology Co., Ltd.	2	20.99	41.98
Project Box (6.2" x 3.54" x 2.3")	B07TS6RY85	Pinfox Tech	2	8.99	17.98
uECG	uECG	Ultimate Robotics	1	-	-
Din Connectors	US_CTA001	AuSL	1	11.35	11.35

- *Table I: Bill of Materials. Items in red were purchased but not used. Items in yellow are items that are not included in the design report.*