Paymodoro Design Report

Austin Lin, David Wang, and Lev Stambler

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of gamifying Pomodoro. Paymadoro 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 Paymadoro.**

***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, 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 minutes 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 Paymadaro. Paymadaro is a system which can be used to monitor a user's Pomodoro sessions and pay them out or penalize them. Our algorithm determines that they remained focused for the span of the Pomodoro session. Paymadoro 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 Paymadoro 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 at least 95% of trials. So, if we place Paymadoro 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 a Pomodoro session be small. This is so that a user is not scared to use Paymadoro and so that Paymadoro remains playful while still being incentivized. 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 Paymadoro intentionally remains focused, Paymadoro should have an error rate of less than 10%. If a user tests Paymadoro, 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 smart contract

The block diagram relating all the components can be found in Fig. 1. The accelerometer and ECG sensor interface solely with the Arduino. The Arduino then interfaces via bluetooth with the desktop app. The desktop app then collects microphone data and interfaces with the blockchain.

We connect the ECG sensor to the user's chest as seen in Fig. 2 and have the accelerometer be directly connected to the

Arduino inside a device that will be clipped to the user's belt as seen in Fig. 3.

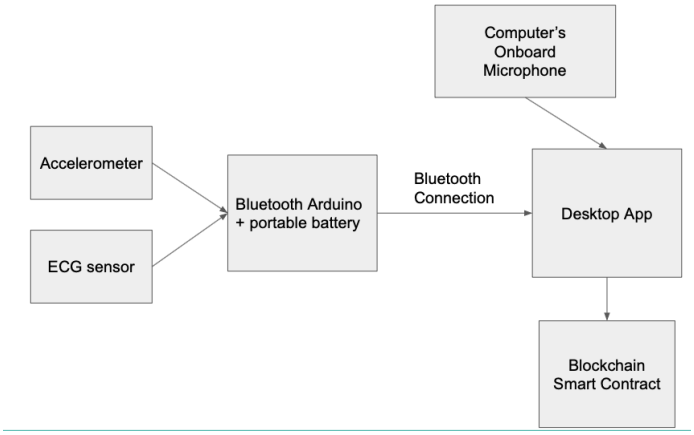

Fig. 1. Our block diagram of different subsystems.

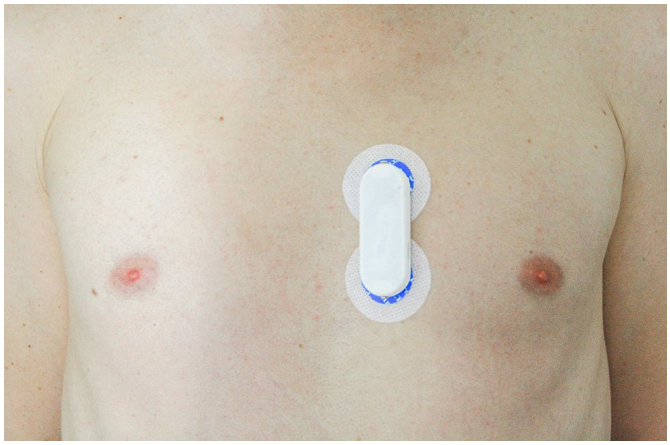

Fig. 2. How the ECG device connects to the user. Note that it must go on bare skin underneath a shirt. https://cdn.tindiemedia.com/images/resize/Xu_xG60E9HECe_v-Eg8DVsQvSZs=/p/fit-in/1370x912/filters:fill(fff)/i/797371/products/2021-07-29T11%3A30%3A37.180Z-uecg-worn-on-chest-900-height.jpg?1627533274

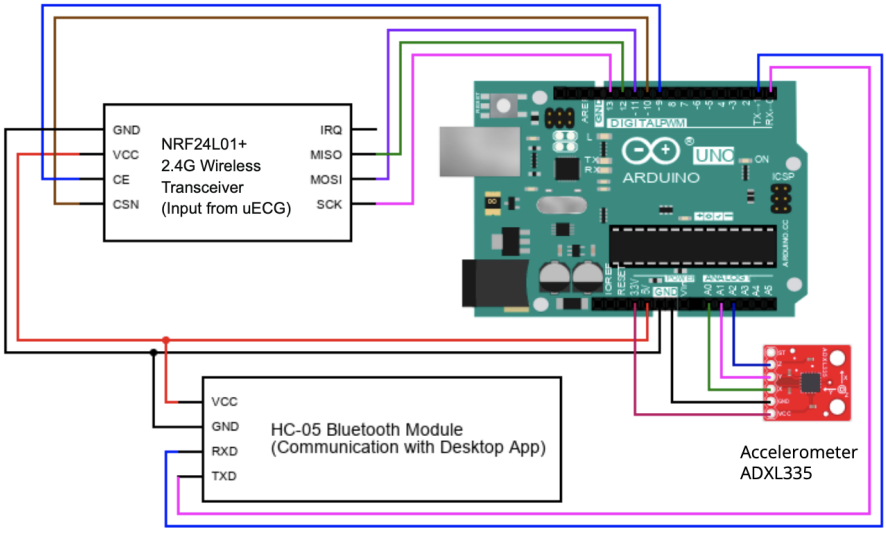

Fig. 3. Wiring diagram for the components inside the belt clipped device. Inside the belt-clipped device there will be a wireless receiver for the uECG device, an accelerometer, and a bluetooth module to communicate with the desktop app.

IV. Design Requirements

A. *Algorithm Requirements*

- **Microphone**: The noise level exceeds the calibration noise level by 10 dB for the entire 15 seconds time interval, we consider this a failure. We choose 10 dB as 10 dB relates to the difference in noise level of a normal conversation, 60 dB, and a vacuum running, 70dB. Thus, we consider 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. **Accelerometer**: Average acceleration exceeds 0.2g for more than 5 seconds. According to Researchgate, 0.2g is the acceleration a person experiences when they start to walk. Thus, if a person is experiencing 0.2g for more than 5 seconds, we assume that they are starting to walk. We consider moving around and changing location as a focus breaking activity. **Heart Rate**

Sensor: Average heart rate is more than 40% above the calibration heart rate for the 15 second period. 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 to a break in focus.

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 only lose 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 rewards from the smart contract and display the amount of tokens earned to the user.

D. *Hardware Requirements*

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

The arduino will continuously collect measurements from the uECG sensor via a 2.4 GHz wireless transceiver and from the accelerometer. Every fifteen seconds, the arduino will take the average heart rate over the last 15 seconds from the uECG data and determine whether the user experienced an acceleration greater than 0.2G for more than 5 seconds. 0.2G is the acceleration that a person experiences during the first 5 seconds of walking. This 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 15 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-C.

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. Other factors we considered are potentially EKG 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 15 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, external 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 really rich user interface and a good looking desktop application, but rather the algorithm and hardware. Therefore, we decided to go with Electron App with 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. Every 15 seconds, the Arduino will send sensor measurements to the desktop app. The app will then store the sensor measurements for each period. At the end of each 15 second cycle, 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 or if the user failed 25% 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

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

- Start Pomodoro Session: starts a Pomodoro session and adds the caler 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 **contract confiscated balance** increases by lock amount.

C. Desktop App

The Desktop App's main purpose is to serve as a communication platform between the blockchain and the hardware. It's 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. We will implement a checksum using CRC-8-Bluetooth for the signals being transmitted to ensure data integrity.

To accomplish our secondary focus, we will utilize an iterative design process to verify what the user needs to see on the user interface. We will first create lo-fi prototypes and conduct user testing to see what type of information is the most important part to the user, and iterate 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 uECG chest mounted electrode. The user's acceleration will be monitored by an accelerometer in the belt clipped device. Finally, the environment sound level will be detected by the computer's onboard microphone.

The arduino will continuously collect measurements from the uECG sensor via a 2.4 GHz wireless transceiver and from the accelerometer directly. Every fifteen seconds, the arduino will take the average heart rate over the last 15 seconds from the uECG data and determine whether the user experienced an acceleration greater than 0.2G for more than 5 seconds. This 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.

The bit stream will be 12 bits long. The first 8 bits will encode the binary representation of the user's average heart rate over a 15 second period. The next 8 bits will encode whether the user experienced an acceleration greater than 0.2G for a 5 second interval. A logical true for excessive movement will be encoded as 01010101 and a logical false will be encoded as 00100100. The last 8 bits will be a cyclic redundancy code calculated using CRC-8-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-C.

VII. Test, Verification and Validation

We will include multiple test suites to verify the design and technical aspects of Paymadoro. First, we want to ensure that session start and end time take no more than 15 seconds. We will test this by starting and stopping sessions 10 times and ensuring that we remain under 15 second latency.

We also want to test our criteria for failure. We will do this by purposefully failing each zone. So, to test that a failure is triggered when any one criteria is failed for more than 25% of the time, we will bring the system into a noisy environment, where a vacuum is running, for 7 minutes and ensure that the session fails. We will also attach the device to a user and get them to start running for 7 minutes and ensure that the session fails. Moreover, we will get a user to start walking and stop walking for the 25 minutes and ensure that the acceleration criteria fails. We will then test that a majority of criteria failing triggers a failed session. We will do this with the following scenarios:

- moving around and chatting
- doing 10 pushups and playing music out loud
- jogging to the kitchen and back, failing the heart rate and acceleration criteria

We further want to ensure that Paymadoro is a good system for incentivizing users to remain in a focused environment. Thus, we want to ensure that users agree with our algorithm's assessments. So we will have 10 users trial run our application. We will then see if the users agree with the app's assessments. As per our use case requirements, we are aiming for < 20% of false positives (where the user is said to be successful, though they believe that they were not), and <10% false negative rate.

Moreover, we also plan on doing accuracy tests for our sensors. These will consist of checking the readout of our sensors against a baseline. So, for our heart rate detection, we will compare the measurements to that of a pulse oximeter for 10 samples. For our microphone, we will compare the measurements to that of an iPhone's microphone for 10 samples. For the accelerometer, we will compare our measurements to that of the iPhone's accelerometer via the Accelerometer App for 10 samples, where each sample consists of starting to walk. In all three cases, we hope that our sensor's accuracy remains within 5% of the baseline measurements as per our design requirements.

Finally, we will test that the smart contract does not extract more than 10 cents of fees by starting and stopping a Pomodoro session and ensuring that the net amount of fees taken does not exceed 10 cents.

In all of our test cases, we will not be using real money. Instead, we will be Near Protocol's testnet. Essentially, a blockchain's testnet works exactly like the real blockchain, but all money is fake and non redeemable for FIAT currency. Thus, we can test without having to expend a lot of unnecessary funds.

VIII. Project Management

A. Schedule

Our current schedule is outlined in Fig. 4. We plan to have the belt clipped device and desktop application coded up integrated on April 1st. We hope to have our entire system integrated on April 22nd.

B. Team Member Responsibilities

Austin will work on implementing the hardware and code for the belt clipped device. David will work on implementing the desktop application. Lev will work on the smart contracts. The entire team will work on integrating the systems together.

C. Bill of Materials and Budget

See Table I for the bill of materials and budget.

D. Risk Mitigation Plans

The most significant risk we have identified is falling behind schedule due to implementation setbacks. The main risk is in sensing and integration. We do not have experience writing code to acquire data from the sensors that we have chosen. Based on our research, it seems that this will be relatively smooth. Integration also carries some risk. It is often very difficult to establish communication between different systems. We have accounted for delays due to implementation and integration setbacks by making the timeline for task completion to be longer than we estimate the task will actually take. If we finish a task early, we will start on the next task. If a task is delayed, the delay will hopefully not exceed our built in slack time.

IX. 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. Pomodor 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.

X. Summary

In summary, our design utilizes hardware sensors to determine a focus score which will allow the platform to justify an incentive system to either reward or penalize the user for their focus session. This creates an incentive on top of simply focusing on tasks for users to utilize the Pomodoro practice. On top of that, the platform as a stakeholder gains the community of users who are willing to use this decentralized system. A few challenges that we will face will be to calibrate the system such that we can minimize false negatives. False negatives are our major concern because it results in a penalty to the user. And if there is a high ratio of false negatives, it will disincentivize the user to use our system as they are getting penalized for doing something correctly. False positives, though less important, is still an important consideration because we wouldn't want the users to figure out a loophole and gain free tokens.

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/.

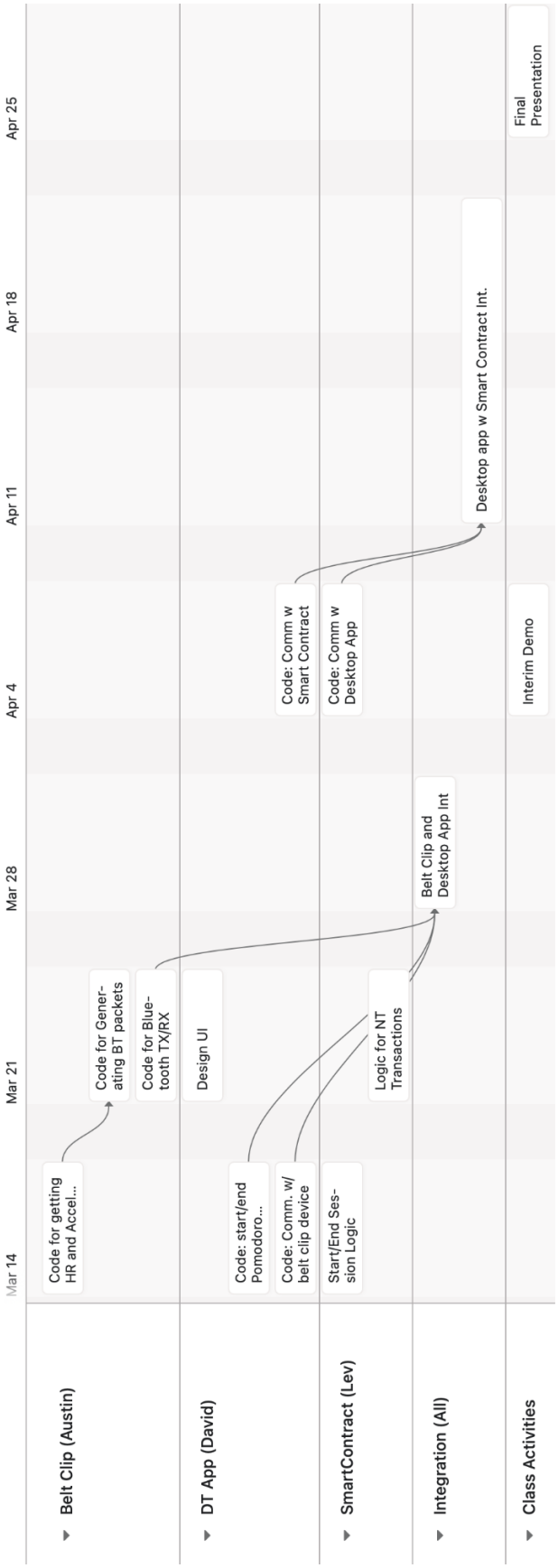

Fig. 4. Project Management Schedule along with important class deadlines

Description	Model #	Manufacturer	Quantity	Cost	Total
Accelerometer	ADXL335	SparkFun Electronics	2	14.95	42.89
2.4 GHz Wireless Transceiver	NRF24L01+	HiLetGo	1	7.89	7.89
Bluetooth Module	8541554474	HiLetGo	1	8.59	8.59
uECG	uECG	Ultimate Robotics	1	0	0
Arduino Uno Rev3	A000066	Arduino	1	0	0

Table I: Bill of Materials