

# Procrastination Tracker and Goal Reminder

Maneesh Kumar Singh (mksingh4@illinois.edu)

## Introduction

As part of final project for CS498 IoT course, we have built a time management system. The system consists of two parts: a social media usage tracker with deterrent and a live display of current tasks or goals in hand using Raspberry Pi. We used AWS to host our IoT infrastructure and thus keeping the cost low.

## Resources

- GitHub Repository <https://github.com/mannbiher/iot-time-management>
- Video URL <https://uofi.box.com/s/hw6nnz6az4d38klui68at7gntorrwbnf>. Video URL is only accessible using UIUC net id.

## Motivation

This semester due to high workload at both office and in MCS program, I was reminded of my non-existent time management skills. Most of the people can relate to having bad time management. So, I decided to build something that could help me manage time and hopefully would be useful for people like me.

When I started tracking my time, I noticed most of my time was spent on unnecessary stuff that was totally unrelated to my work. I was surprised to see this is common psychological phenomena. This phenomenon is called procrastination and it has been defined as the action of delaying or postponing tasks which may have negative consequences if not completed. A lot of study and research has been done on procrastination. I was surprised to see how common it is among the population. Some studies shows that the 20-25% of general population suffers from procrastination [1]. In academia, this figure is higher and, in a study, up-to 70% of students self-identified them as procrastinators [2]. Among these students 50% procrastinated consistently and considered it to affect their lives negatively [3].

It has also been shown to negatively affect one's health, self-esteem, and results in poor academic performance [4]. Due to its larger prevalence and psychological effects, identifying the reasons for procrastination and methods to reduce procrastination have also been researched. Several psychologists have attributed it as a coping mechanism or avoidance behavior to feel good by not stressing to do the important tasks [5].

Below I provide summary from scientific literature for some of the reasons for procrastination.

1. Instance gratification and immediate urgency of managing negative moods [6].

People procrastinate to avoid the negative feeling. Instead of facing the stress to work on most important thing, it is always easy to open video game or browse social media. The more, they are near to deadline, the more they are stressed and feel urge to procrastinate more [7].

2. Disconnect between present and future self [8].

Research suggests that people do not consider their future self to be the same person and thus prefer to procrastinate assuming the consequences would be borne by their future self.

3. When a task is too difficult, we are more likely to put it off as much as possible [9]

Similarly, below I provide summary of actions one can take to reduce procrastination. The blogs by James Clear [10] and articles by Ana Swanson [7], and Charlotte Lieberman [11] summarizes methods to stop or reduce procrastinating very nicely.

1. Forgive yourself for procrastinating.

Research has shown that people who forgive themselves for past procrastination tend to do less procrastination in future.

2. Make the rewards of taking actions more immediate.

Bring the benefit of long-term choices more immediate. Do boring things with the one you like. E.g., Only listen to music while you exercise.

3. Make the task more achievable.

Break the task into smaller chunks that can be completed. Do not wait to be in a mood to work on a task. Just get started. Remove every roadblock. E.g., do not keep your cellphone near you when you are studying.

4. Always have a next action. Motivation follows action.

When people do not know what the next action for a large task is, they tend to wander off and unknowingly starts procrastinating.

5. Avoid chronic procrastination with visual cues.

A visual cue is something that you can see and allows you to act. It can be used to overcome procrastination by triggering habits and measure your progress.

6. Make your temptations more inconvenient.

Add friction to your procrastination. Keep a password on your cellphone which is so difficult to remember, so that you do not get distracted by social media. However, what I found that self-control in using social media is exceedingly difficult. If a mentor, spouse, or friend could help you by notifying you when they think you are overdoing things would really help.

The first three solutions are different for each person and require personal planning and implementation. The last three solutions are generic and are applicable to most people. As part of this project, we worked on building a system which can help people to have next set of action, visual cue for progress and social media usage tracker and deterrent. To keep the system design simple, we have broken down these solutions into two different parts.

### [Social Media usage tracker and deterrent](#)

we have built a system that tracks user cellphone usage and notifies a person (self or other) when the user overuses the social media.

## Goal Tracker (Next set of action and visual cue for progress)

One of the important steps for better time management is to have your goal or tasks in front of you. Everyone has some goals that they want to achieve. As a software developer, one of the goals that I am currently pursuing is to solve 5 LeetCode<sup>1</sup> questions every day. It would be nice to display the stats of my progress and keep it in front of me. This could be generalized to show progress from any web service or application.

We also implemented a naïve spaced repetition algorithm to give end user problem based on last submission. Spaced repetition is a learning paradigm where more difficult and newly introduced problems are shown more frequently while older and less difficult problems are shown less frequent. It has been proven to increase rate of learning [12].

## Technical Approach

### Raspberry Pi Touch Screen

For this project, we used Raspberry Pi touch screen for display of progress tracker. We connected it to Raspberry Pi following below circuit diagram. Power and Ground cables from touch screen PCB are connected to GPIO pin 4 and 6, respectively. DSI cable is connected between PCB and Raspberry Pi display DSI connector.

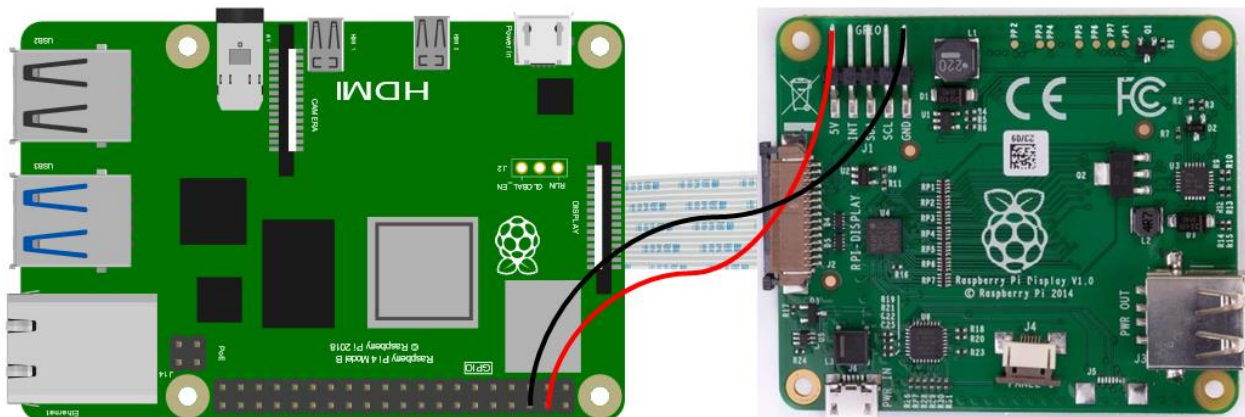


Figure 1 Touch Screen Circuit Diagram

### Social Media usage tracker and deterrent

We use Raspberry Pi as network monitoring device. In-order to allow all internet traffic from user device to flow through it, we set it up as a wireless access point and then configure user cellphone to connect to new Wi-Fi network. We use tcpdump utility to monitor the network and use python modules to send the captured packet data to AWS IoT using MQTT protocol. We setup rules to fetch this data to AWS IoT analytics for later analysis and simultaneously send it to AWS Lambda for immediate action. AWS Simple Notification Service (SNS) is used to send message to another user device. The architecture is shown in Figure 2.

---

<sup>1</sup> [LeetCode.com](https://leetcode.com) is an interview question preparation website.

Below steps provide an overview of data flow.

1. Raspberry Pi is setup as a wireless access point creating a new Wi-Fi network.
2. Pi also acts as network router and performs network address translation to allow devices in new network to access Internet.
3. tcpdump captures internet traffic on Wi-Fi interface and writes to pcap files.
4. Python module parses output pcap files from tcpdump. It transforms the capture by applying Reverse DNS lookup, concatenating multiple records into one and then publishing JSON data to an IoT topic.
5. AWS rule is setup on the IoT topic to send the message to both AWS Lambda and AWS IoT analytics.
6. AWS Lambda contains a list of blacklisted social networking websites (E.g., 9gag.com, facebook.com).
7. It checks if the usage of any blacklisted domains is over a threshold. If yes, it publishes to an SNS topic.
8. Mentor/Spouse/Well-wisher/User is subscribed to SNS topic for SMS delivery mode.
9. Concerned person gets a SMS detailing which websites are overused.
10. AWS IoT analytics channel captures the messages, process it through pipeline and saves it in datastore.
11. Now the data can be used for visualization, analytics, and machine learning to further help in time management.

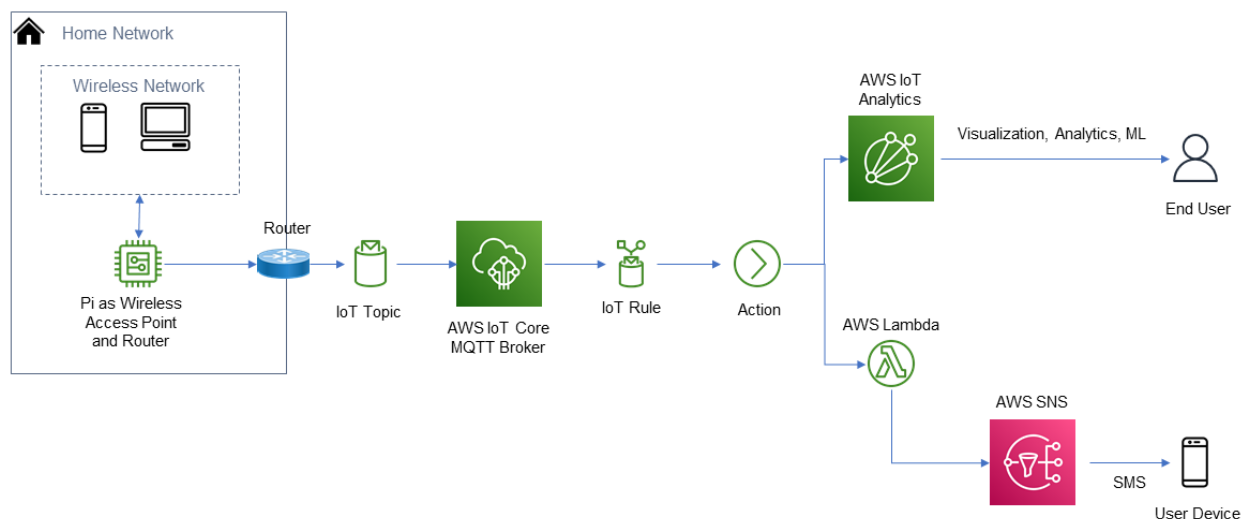


Figure 2 Social Media Usage Tracker and Deterrent

## Goal Tracker

*Keep your goals in sight.*

For this part of the project, we chose to show LeetCode progress status and a problem counter for the day. AWS Lambda is scheduled to run on an interval. It fetches the progress information from LeetCode and publishes that to a topic. Raspberry Pi is connected to a touch screen

display. We developed an Electron app to show the current progress and to allow user to fetch new updates. Electron app is subscribed to the topic and gets progress updates from Lambda. It also publishes to a different topic. An AWS IoT rule is setup to trigger LeetCode lambda on receiving message from the topic. We use two different topics to allow two-way communication.

As shown in Figure 3, below steps provide an overview of the data flow.

1. AWS CloudWatch event rule to trigger AWS Lambda periodically
2. AWS Lambda pulls the statistics for user from LeetCode web application. It applies logic to find most suitable problems to be solved based on Spaced Repetition technique and publishes both the current progress stats and problems for today to the topic.
3. An Electron Application subscribes to AWS IoT topic when it starts.
4. The app updates the UI, whenever updated information is received on the topic.
5. The app also publishes to a different topic whenever update is requested by user.
6. IoT rule is setup on the topic to trigger same lambda function.
7. Lambda receives the event through IoT action and then fetches new stats from LeetCode and update it based on topic message. It then sends new information back to the specified topic.

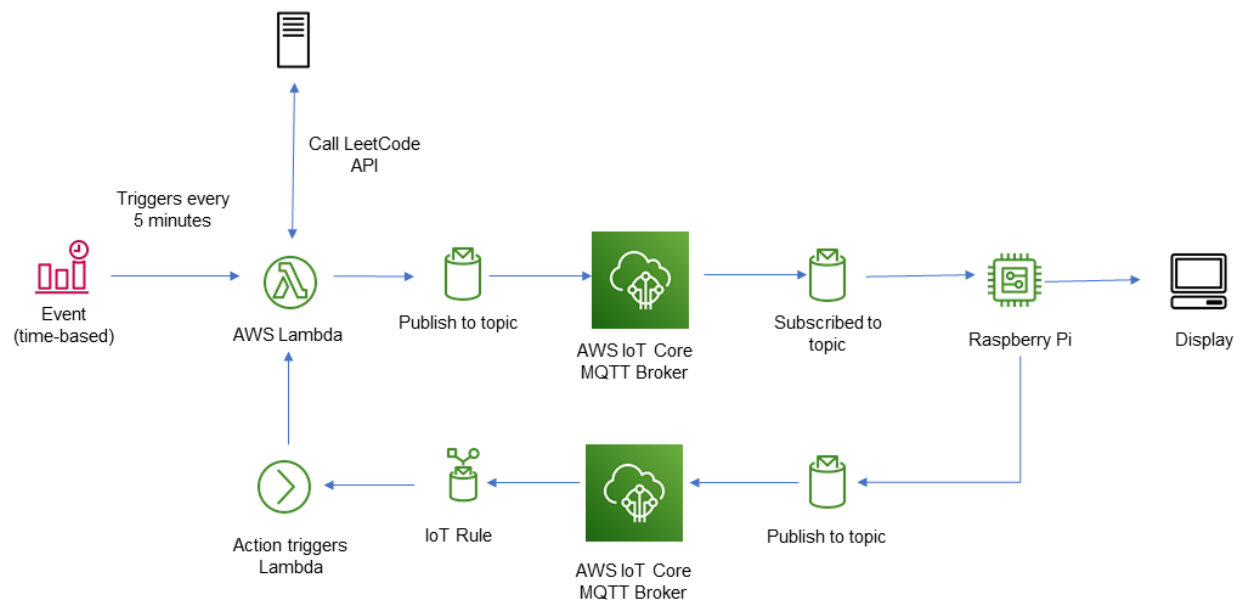


Figure 3 Goal Tracker

## Implementation Details

### Hardware Components

To successfully setup the system, an internet router with available ethernet port is required. A mobile phone is required to test the usage tracking and notification. In addition, below parts/devices are required to complete this project.

1. Raspberry Pi 4B (8GB Model preferred)
2. USB-C Cable for Powering Pi
3. Ethernet Cable
4. Raspberry Pi Touch Display

### Assemble Raspberry Pi 4B with Touch Display.

Assembling Raspberry Pi with touch display was easy. We were able to assemble it in minutes by following [YouTube video](#) tutorial by ETA Prime [13].

In Figure 4, we show the colored wires and DSI ribbon cable connected to PCB board of touch screen display. We learnt that we only need to connect 5V and GND cables (Red and Black respectively) to Raspberry Pi GPIO pin 4 and 6. However, we connected all the cables as done in the video tutorial.

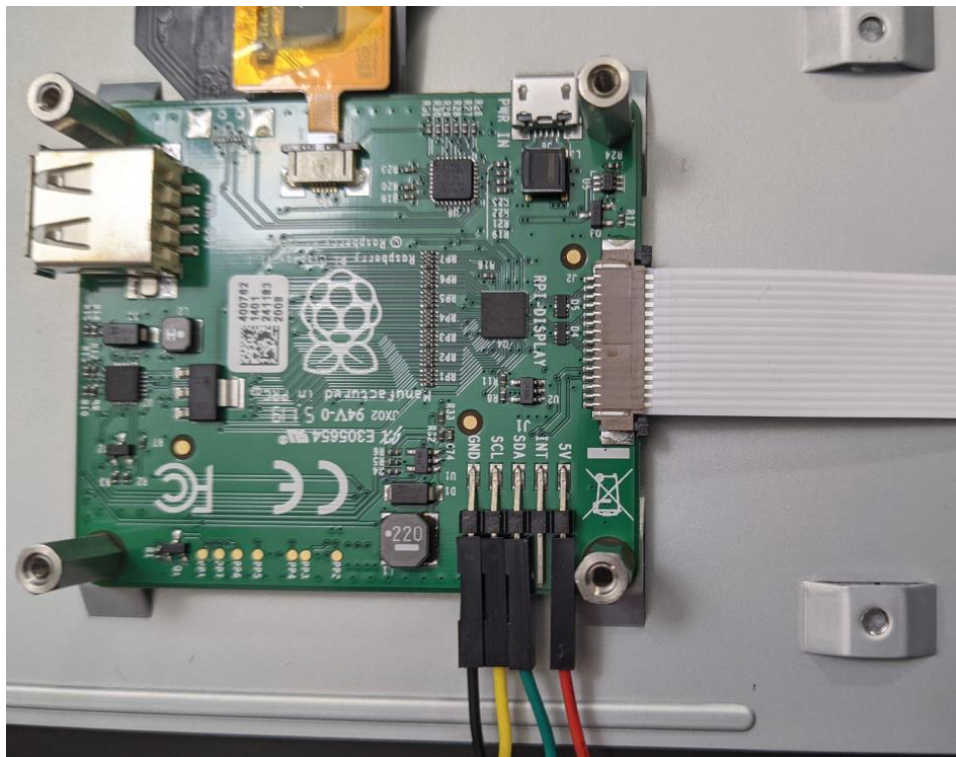


Figure 4 Touch Screen PCI Board



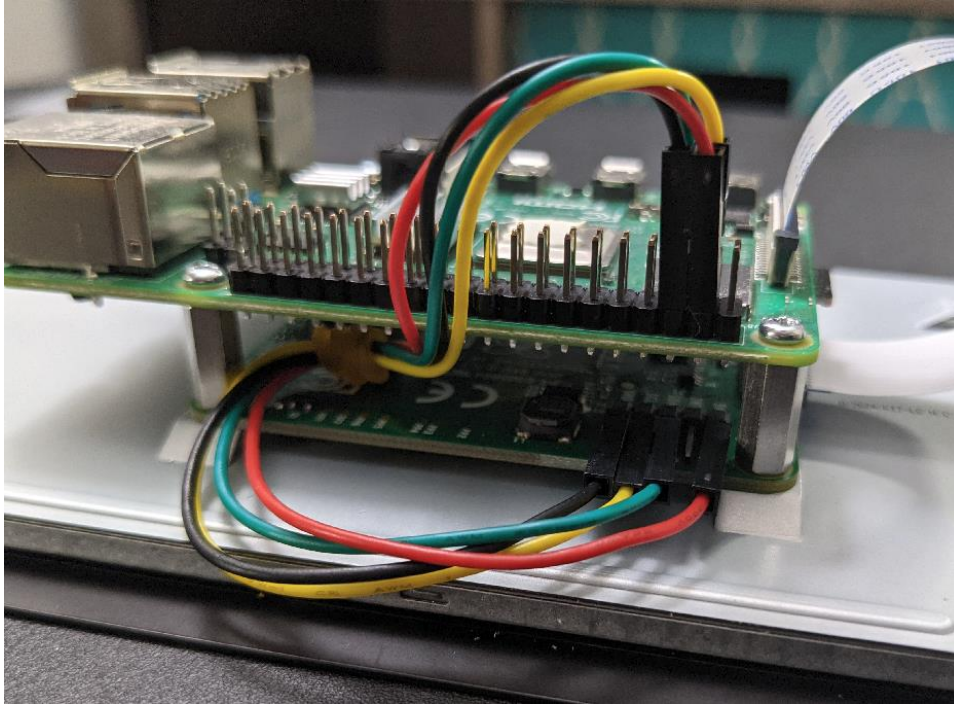


Figure 5 Fully Assembled Touch Screen

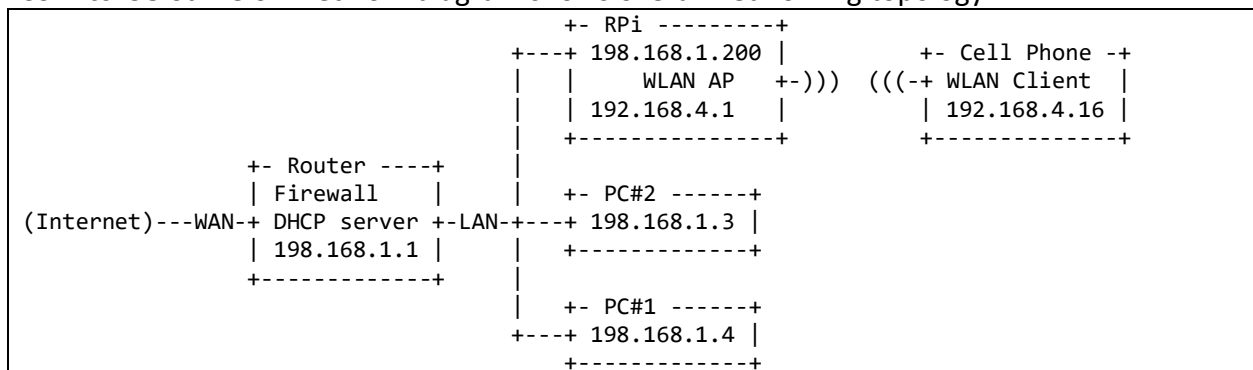
## Social Media Usage Tracker and Deterrent

### Raspberry Pi as Wireless Access Point

First thing we need to track social media usage is to have all the traffic from intended device to flow through a monitoring device. We chose Raspberry Pi for that. There are two ways to put Raspberry Pi between devices and internet router.

1. Setting up Raspberry Pi as routed wireless access point [14]  
This creates a new wireless network which is completely managed by Pi.
2. Setting up Raspberry Pi as bridged wireless access point [15] within an existing ethernet network. In this mode Raspberry Pi extends existing network.

We chose the first one for its simplicity and giving us complete control over Wi-Fi network. We followed the Raspberry Pi documentation and were able to create a new wireless network with ESSID cs498iot. Below network diagram shows overall networking topology.



As the tutorial is quite simple and we did not have to change anything except the wireless access point setting, so we will just mention our changes from the official tutorial. The official tutorial uses GB as country code and 2.4 GHz band, which can be changed based on your network. Here is the setting for us for `/etc/hostapd/hostapd.conf`. We are using 2.4 GHz band on channel 6 and our country code is US.

```
pi@raspberrypi:~  
country_code=US  
interface=wlan0  
ssid=cs498iot  
hw_mode=g  
channel=6  
macaddr_acl=0  
auth_algs=1  
ignore_broadcast_ssid=0  
wpa=2  
wpa_passphrase=XXXXXXXXXX  
wpa_key_mgmt=WPA-PSK  
wpa_pairwise=TKIP  
rsn_pairwise=CCMP  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

Figure 6 `/etc/hostapd/hostapd.conf`

We also enabled masquerading for wireless interface, so that our devices can talk to internet. After setting up Raspberry Pi as a wireless access point, we were able to connect an android device to the Wi-Fi network and test its connectivity to Internet.

### tcpdump

After having the network traffic flow through Pi, now we needed a traffic capture utility like Wireshark. We chose tcpdump for this. By default, tcpdump does not run with normal user permissions, you need to run it using sudo. So, we created new group and provided it permission to run tcpdump and then added our username to be able to run it as non-root user [18].

```
pi@raspberrypi:~ $ sudo groupadd pcap  
pi@raspberrypi:~ $ sudo usermod -a -G pcap $USER  
pi@raspberrypi:~ $ sudo chgrp pcap /usr/sbin/tcpdump  
pi@raspberrypi:~ $ sudo chmod 750 /usr/sbin/tcpdump  
pi@raspberrypi:~ $ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump  
pi@raspberrypi:~ $
```



We setup a script to run tcpdump with below parameters.

```
#!/usr/bin/env bash
mkdir -p /var/tmp/pcap
tcpdump -i wlan0 -nlrt '(port 443 and tcp and greater 128) or (port 53 and udp)' -W 100 -G 60 -w
/var/tmp/pcap/%Y%m%d%H%M%S.pcap
```

The first command creates storage directory for pcap files if it does not exist already.

Parameters provided for tcpdump are:

- -i wlan0 interface. We are listening to wlan0 interface which is used as an access point. By listening to this interface, we would not have to deal with data that is transferred from Raspberry Pi itself.
- -n Disables the reverse DNS lookup on each capture. It makes the capture process much faster.
- -tt Provides timestamp in human readable format
- -l enable buffers for console output
- -W 100 Write only 100 files. Tcpdump will exit once it writes more than 100 files. This is required, as we do not want tcpdump to exhaust all space on Pi.
- -G 60 we want tcpdump to write one file every minute. This file is parsed and delete by python module and thus keeping the number of files in balance.
- -w /var/tmp/pcap/%Y%m%d%H%M%S.pcap is the naming format of pcap files using [strftime](#) format.
- (port 443 and tcp and greater 128) or (port 53 and udp) We capture all traffic on port 443 for TCP protocol where package size is greater than 128 Bytes. Mostly all websites use port 443 for HTTPS so it is a good filter for us to concentrate only on subset of packets. We also filter traffic on port-53 and UDP protocol. This captures DNS traffic.

### Python modules

We run our python module (mqtt\_publish.py) at the same time when tcpdump is running. This module reads the pcap file written by tcpdump and parses them using scapy python package. We use a [python function written by Je Clark](#) [18] to parse DNS information from pcap records and use it to build a dictionary for reverse DNS lookup. We do this to determine the hostname and domain for each packet. This is required to apply domain-based filtering done in later stages. Instead of parsing both to and from connection to our device, we consider only the incoming connections. For each connection, we concatenate the size with the previous record if current source IP, destination IP, source port and destination port are same as previous record. We do this to reduce the amount of data that we send to AWS IoT. After summarizing all data records for a pcap file, we publish this information in batches of JSON array each containing 500 records to AWS IoT topic. We do this to avoid crossing AWS IoT per message size limit which is currently 128 KB.

## AWS IoT Core

In AWS IoT core we setup a rule to send the message to both AWS IoT Analytics channel and AWS Lambda. Python module publishes messages to topic `cs498/time/usage`.

### Rule query statement

[Edit](#)

The source of the messages you want to process with this rule.

```
SELECT * FROM 'cs498/time/usage'
```

Using SQL version 2016-03-23

### Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Send a message to a Lambda fu...  
CS498\_NotifyMentor

[Remove](#)[Edit](#) ▶

Send a message to IoT Analytics  
cs498\_time\_channel

[Remove](#)[Edit](#) ▶[Add action](#)

## AWS IoT Analytics


This was not in scope for the proposal. Professor Matt in his proposal acceptance mail, mentioned that it would be interesting to incorporate analytics or machine learning in the project. And after lab 4, we knew that it can be done easily using AWS IoT analytics service. We decided to incorporate it to show some basic visualization and as proof of concept for future machine learning tasks on the collected data.

We create an AWS IoT analytics channel, data source, a pipeline connecting channel to data source. We use a lambda to flatten the records in the message. We created a dataset to query this data.

### *FlattenRecords Lambda*

We are sending a JSON array as a message to AWS IoT analytics. Currently AWS IoT SQL does not support flattening JSON arrays to individual records. The JSON array arrives at AWS IoT Analytics as an array of array which cannot be queried efficiently. So, we had to write this simple lambda to flatten the array. This lambda is setup as an Activity in the IoT Analytics

pipeline process. The source code has been taken from an [AWS forum recommendation](#) by AWS support person [18].

 Transform message with Lambda function TRANSFORM

### Incoming messages

No attributes available

### Transform message with Lambda function

Use a Lambda function to process or enrich your message. AWS IoT Analytics batches the messages before sending to Lambda function.

**Lambda function**  
CS498\_FlattenRecords

**Batch size**  
1

### Outgoing message

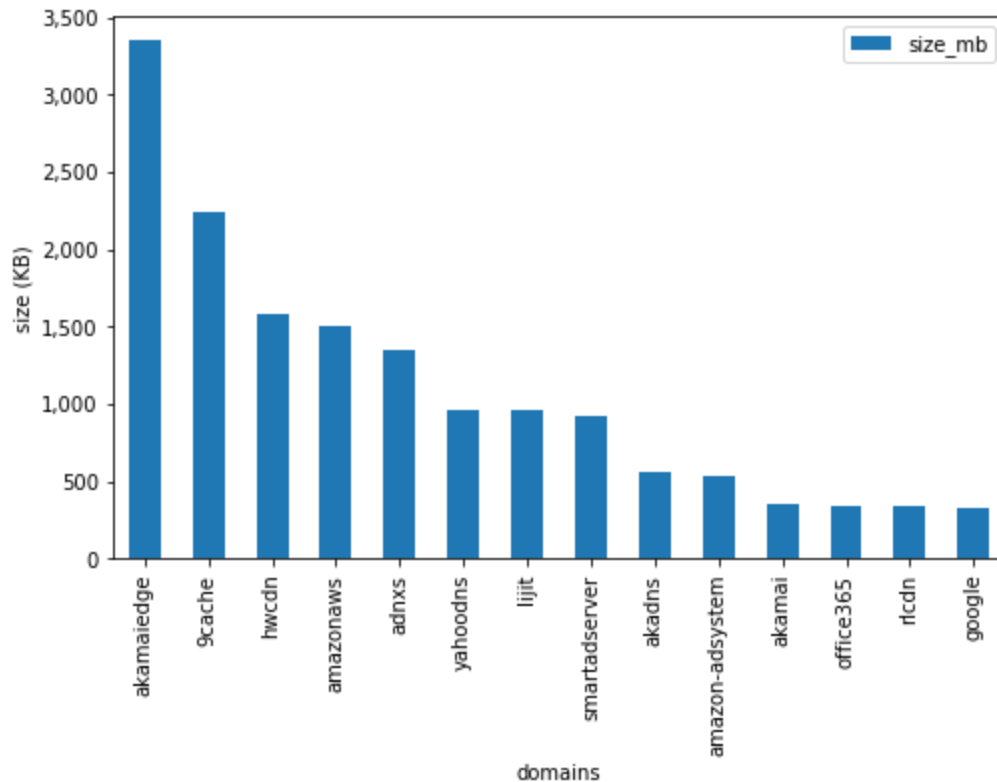
Below is a list of attributes to be included in outgoing messages.

In order to execute AWS lambda from pipeline, we also had to add a resource level policy to allows IoT Analytics service to invoke the lambda function.

```
mann@mann-G55VW:~$ aws-vault exec mann -- aws lambda add-permission --function-name CS498_FlattenRecords --action lambda:InvokeFunction --statement-id iotanalytics --principal iotanalytics.amazonaws.com
{
  "Statement": "{\n\"Sid\": \"iotanalytics\", \"Effect\": \"Allow\", \"Principal\": {\n\"Service\": \"iotanalytics.amazonaws.com\"}, \"Action\": \"lambda:InvokeFunction\", \"Resource\": \"arn:aws:lambda:us-east-2:057733667042:function:CS498_FlattenRecords\"}"
}
```

### *IoT Analytics Notebook*

We use AWS IoT analytics notebook to work on dataset created using query dataset. Below we show the visualization of usage of top domains based on size. Once the data is in IoT datastore we can create multiple query datasets and use them for analytics or machine learning.



### NotifyMentor Lambda

This lambda function is triggered when messages arrive in topic `cs498/time/usage`. The lambda function maintains a global dictionary which keeps track of usage size for each domain so far. It also maintains a list of blacklisted websites and usage limit. On each invocation, lambda adds the new data size to existing dictionary for blacklisted sites. Then it checks if any of the domains are over the usage limit. If they are, the lambda sends notification to AWS SNS, which in turns sends SMS message to people subscribed to the topic. It also deletes the current entry for the domain, so that it does not send repeated messages for every next revocation.

As AWS does not guarantee runtime reuse for each lambda invocation, the dictionary may be lost or not shared with other instances. However, this simple design is good for our use case.

The AWS Lambda is provided a role with SNS publish access for it to publish messages to SNS topic.

### Goal Tracker

Goal tracker system allows user to see the current progress on LeetCode and provides him the next problem to solve based on a naïve spaced repetition algorithm. We list detailed description for each component and data flow among them.

## LeetCode

LeetCode is an interview question preparation website. It hosts programming questions in multiple categories. E.g., algorithms, database. It also maintains a hardness level of a problem which could be easy, medium, and High. Every problem is assigned to one of the hardness levels. We make two LeetCode API calls: Pull current progress status for the user and then all the problems user has tried or not tried.

## AWS CloudWatch Event Rule

AWS CloudWatch Events is a service which provides capability to connect and handle system events provided by other AWS service or custom events. We can use rules to match events and route them to targets. This is similar to AWS IoT Core rules. Using CloudWatch events we can setup rule to trigger AWS Lambda function on a schedule. Using this functionality we setup LeetCodeNotification lambda to run every five minutes.

### Rules > cs498\_leetcode\_rule

#### Summary

**ARN** ⓘ `arn:aws:events:us-east-2:057733667042:rule/cs498_leetcode_rule`

**Schedule** Fixed rate of 5 minutes

**Status** Disabled

**Description** Get Leetcode stats every five minutes and send to IoT topic

**Monitoring** [Show metrics for the rule](#)

#### Targets

Filter: <input type="text"/>			
Type	Name	Input	Role
Lambda function	<a href="#">CS498_LeetCodeNotification</a>	Matched event	

## LeetCodeNotification Lambda function

This lambda function is responsible for sending current user progress and next problem to an IoT topic. In this lambda function, we wrote a thin wrapper client to call LeetCode API and get the current progress for the user. We also get metadata of all problems and separate them in done and remaining categories.

One of the problems that I faced while using LeetCode is to not have functionality of Spaced Repetition. There is no way to filter problems by solved date and submission attempts and try them again.

We tried to solve this problem by writing basic algorithm to mix the problems that are presented to the user.

### *Naïve Spaced Repetition*

The ideal way should be to mix new problem with an old problem based on user submission date and number of attempts it took for you to solve it. However, LeetCode does not have an API where you can query problems by submission time or attempts. We must call LeetCode API for each problem in done category and get the timestamp and number of attempts and then chose the most appropriate problem. As number of solved problems could be in hundreds, we had to minimize API calls and keep our system simple. So instead, we chose to get five random problems from done problems. For each of five problems, we do individual queries to get status for these problems and then sort these problems by submission timestamp and send the oldest problem. For remaining problems, we simply chose a problem at random.

### *Today Problem count*

As stated above, it is not possible to query problems by date or submission attempts. So, without this ability, we would not be able to view status of today's goal. Our intention is to have a Today's problem counter which gets reduced each time a problem is solved. To overcome this, we decided to save the state of today's problems in DynamoDB table. DynamoDB is a key-value database provided by AWS. So, each time user solves a problem, he must click Get Next button to publish topic which will have the Id of the problem solved. On trigger based on IoT rule, the lambda would query LeetCode to confirm if the problem is indeed solved today and if yes, stores the entry in DynamoDB and returns the new problem solved count.

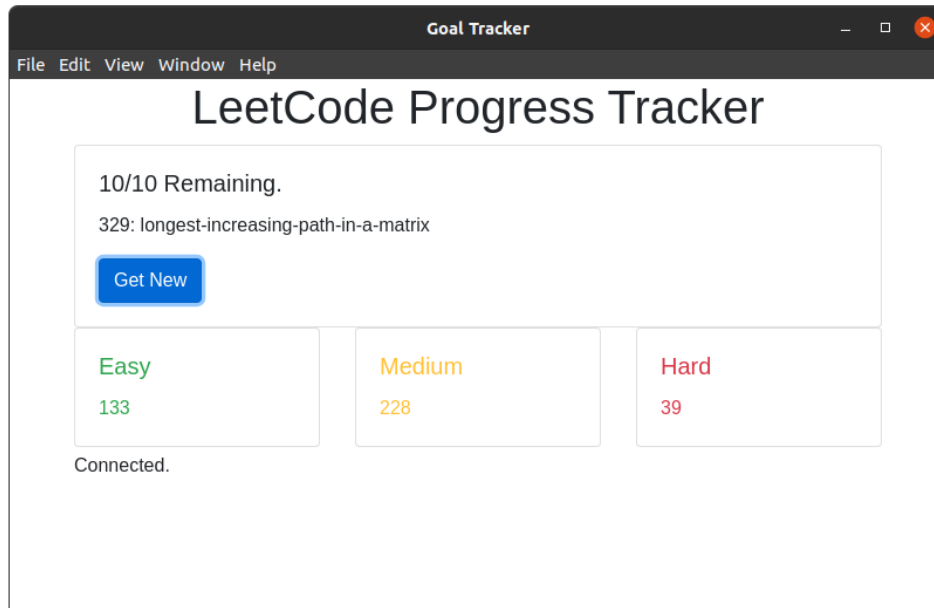
### *Electron App*

We were initially not sure about the UI application to be used on Raspberry Pi to display user progress. As we learnt about Electron application in Lab-2, we decided to use the learning for this project. We created a quite simple Electron application. This application resolution is set to 800 x 480 to match the Raspberry Pi touch screen display resolution. When the application starts, it subscribes to IoT topic `cs498/time/leetcode` which is used by Lambda function to send user progress.

Initially we planned and implemented only one way communication. When Electron application on Pi starts, display remains empty till the lambda runs and publishes message to topic. However, after using it for some days, we realized that we need to have ability to ask for update. For this, we also developed message push from Raspberry Pi to another topic `cs498/time/new` and setup IoT rule to send the event to same lambda function. This allowed us to request information when required. This was helpful when we solved a problem and wanted the Pi screen to refresh immediately.



Application allows user to request update and new problems by clicking on Get New button.



## Results

We were able to implement and successfully test Social media usage and Goal tracker. We get notification for over-usage of social media. We can query and visualize the usage data in AWS IoT analytics. Goal tracker is our personal favorite as it helps us to be on track by displaying our progress and providing us next action to perform.

We enjoyed and learnt a lot during this project execution. We faced below difficulties implementing this project and were able to resolve or work around them.

## Challenges

No connectivity to Pi Access Point

After setting Raspberry Pi as wireless access point after following the documentation, we were not able to connect any device to this Wi-Fi network. After wasting a lot of time going through various forums, we realized that it was due to UFW firewall blocking all connections to closed ports. We disabled UFW for rest of the project as it was difficult to figure out which port would be used by client application when we are running Raspberry Pi as router.

```
mann@mann-G55VW:~$ nmcli --ask dev wifi connect cs498iot
Password: .....
Error: Connection activation failed: (5) IP configuration could not be reserved
(no available address, timeout, etc.).
mann@mann-G55VW:~$
```

```

pi@raspberrypi:~ $ sudo ufw status
Status: active

To Action From
--
22/tcp ALLOW 192.168.1.0/24
65432/tcp ALLOW 192.168.1.0/24
8000/tcp ALLOW 192.168.1.0/24
5900 ALLOW 192.168.1.0/24
8883/tcp ALLOW 192.168.1.0/24

pi@raspberrypi:~ $ sudo ufw disable
Firewall stopped and disabled on system startup
pi@raspberrypi:~ $ sudo ufw status
Status: inactive
pi@raspberrypi:~ $

```

### Reverse DNS Lookup

While doing reverse DNS lookup to populate hostnames, we learnt that reverse DNS does not work like an actual reverse of DNS lookup (e.g., IP address to hostname), but rather depends on a PTR DNS record which may not be present for all IP addresses. Therefore, you may get some hostnames that resolve to an IP address, but when you do reverse DNS lookup you get no result. To solve this problem, we must capture the DNS requests also that happen before the actual HTTP call. We followed this approach and for the most part it worked fine. It only failed when the client did not perform a DNS lookup as it already had a DNS cache entry. It also made us think and research as how the popular network logging and reporting tools work and how do they populate the DNS entry.

### LeetCode Login Change

The whole idea for using spaced repetition technique on LeetCode problems was inspired from various CLI tools that leverage LeetCode undocumented API to automate some of the tasks. I realized early in the project that these tools are outdated and no longer works as LeetCode has changed its login method to involve Google reCAPTCHA which successfully fails all attempts by automated bots to login to LeetCode using username and password. The only work around that works is to login from a Google Chrome browser and copy the cookie content to python module for it to call LeetCode API without error. Also, we found if we copy the cookie value from Firefox, it does not work.

### Future Enhancements

There are multiple ways this system can be enhanced to be more helpful. Currently the amount of data that is sent to AWS IoT is huge. We can compress the messages and send binary messages using gzip compression.

We could install AWS Alexa SDK on Raspberry Pi and make it work like a personal organizer. You could ask questions like "Alexa, what is in for today?" and it would answer all the tasks that you need to do today. You could also ask it to add tasks to your calendar and it can also pick the tasks from your outlook via API.

## Acknowledgements

Goal tracker is my personal favorite and was on my agenda to build something like for a long. I am thankful for Professor Matt to provide me an opportunity to implement this idea while learning about IoT systems and technologies.

I would also like to acknowledge the support from TA and fellow classmates in completing this project and other course labs.

## References

1. Ferrari, J. R., Díaz-Morales, J. F., O'Callaghan, J., Díaz, K., Argumedo, D. (2007). [Frequent behavioral delay tendencies by adults: International prevalence rates of chronic procrastination](#). *Journal of Cross-Cultural Psychology*, 38, 458–464.
2. Schouwenburg, H. C. (2004). [Procrastination in academic settings: General introduction](#). In H. C. Schouwenburg, C. H. Lay, T. A. Pychyl, J. R. Ferrari (Eds.), *Counseling the procrastinator in academic settings* (pp. 3–17). Washington, DC: American Psychology Association.
3. Day, V., Mensink, D., O'Sullivan, M. (2000). [Patterns of academic procrastination](#). *Journal of College Reading and Learning*, 30, 120–134.
4. Duru, Erdinç; Balkis, Murat (2017). [Procrastination, Self-Esteem, Academic Performance, and Well-Being: A Moderated Mediation Model](#). *International Journal of Educational Psychology*, v6 n2 p97-119 Jun 2017
5. Fiore, Neil A (2006). [The Now Habit: A Strategic Program for Overcoming Procrastination and Enjoying Guilt-Free Play](#). New York: Penguin Group. p. 5. [ISBN 978-1-58542-552-5](#)
6. Sirois, F. and Pychyl, T. (2013) [Procrastination and the Priority of Short-Term Mood Regulation: Consequences for Future Self](#). *Social and Personality Psychology Compass*, 7 (2). 115 - 127. ISSN 1751-9004
7. Pychyl, T. (20 February 2012). ["The real reasons you procrastinate — and how to stop"](#). *The Washington Post*. Retrieved 20 February 2012
8. Hal E. Hershfield. [Future self-continuity: how conceptions of the future self transform intertemporal choice](#). First published: 24 October 2011 <https://doi.org/10.1111/j.1749-6632.2011.06201.x>
9. Romeo Vitelli Ph.D. (July 1, 2013) [Getting Around to Procrastination. What causes people to procrastinate? And is it necessarily a bad thing?](#)
10. James Clear. (May 2021). [Procrastination: A Scientific Guide on How to Stop Procrastinating](#)
11. Charlotte Lieberman. (March 25, 2019). [Why You Procrastinate \(It Has Nothing to Do With Self-Control\)](#)
12. Smolen, Paul; Zhang, Yili; Byrne, John H. (January 25, 2016). ["The right time to learn: mechanisms and optimization of spaced learning"](#). *Nature Reviews Neuroscience*. 17 (2):

77–88. arXiv:1606.08370. Bibcode:2016arXiv160608370S. doi:10.1038/nrn.2015.18. PMC 5126970. PMID 26806627.

13. Official Raspberry Pi 4 7" Touchscreen Display Review  
<https://www.youtube.com/watch?v=J69-bxOSMC8>
14. Setting up a Raspberry Pi as routed wireless access point  
<https://www.raspberrypi.org/documentation/configuration/wireless/access-point-routed.md>
15. Setting up a Raspberry Pi as a bridged wireless access point  
<https://www.raspberrypi.org/documentation/configuration/wireless/access-point-bridged.md>
16. DNSDissect script provided by je-clark in his GitHub repository. <https://github.com/je-clark/ScapyDNSDissection>
17. Recommendation on processing attributes with array of values.  
<https://forums.aws.amazon.com/thread.jspa?threadID=288424>
18. Tcpcat permission problem. <https://askubuntu.com/questions/530920/tcpdump-permissions-problem>