Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Team 28:
Smart Caller ID for Landlines
Bi-Weekly Update 2

Amy Chen
Scott Kevil-Yeager
Matthew Hebrado

Sponsor: Dr. Tod Cox

TA: Rohith Kumar

Project Summary

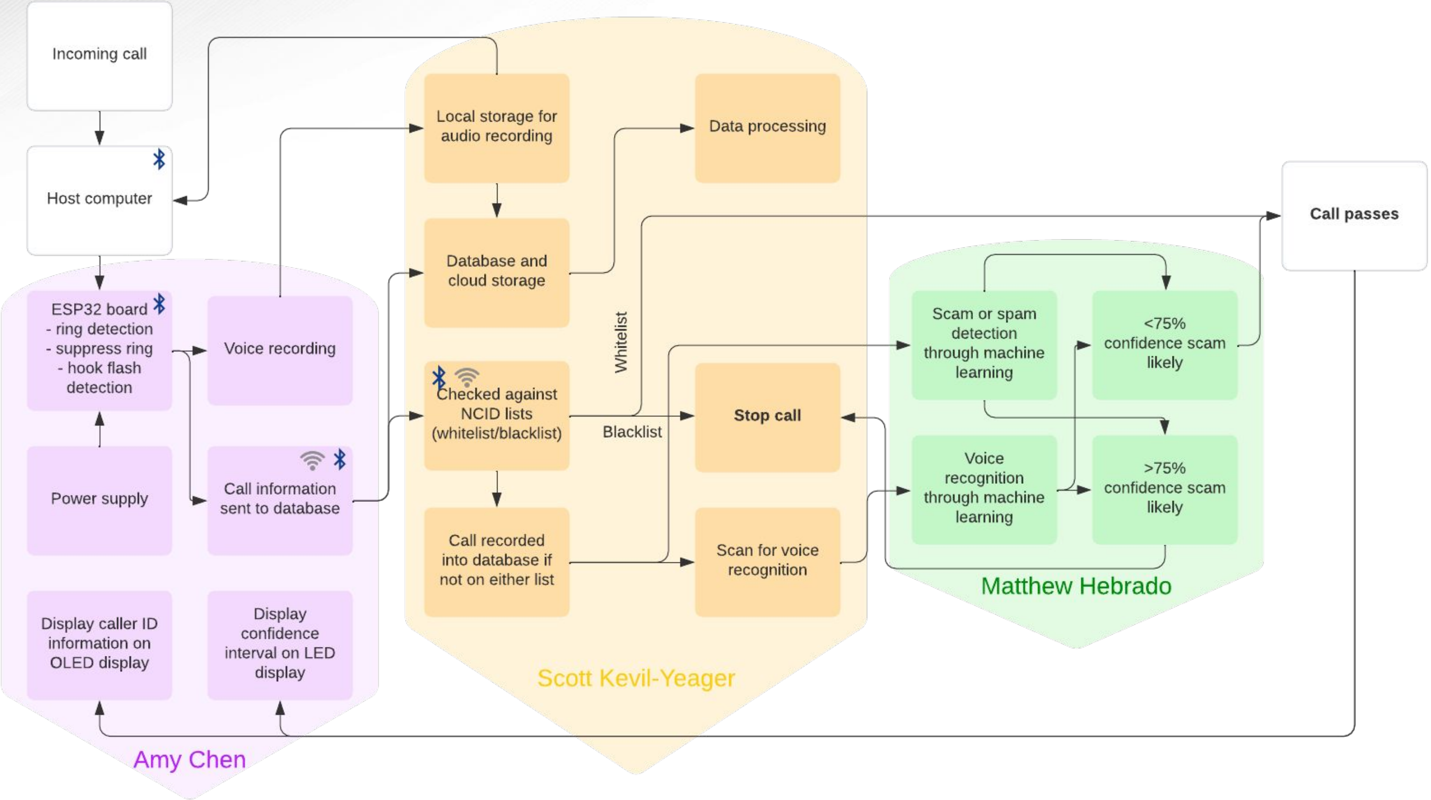
The Problem

- Many older residents in the U.S. still rely on a landline for telephone communications and many times, they are scammed out of money. This process can go on for months before someone notices.

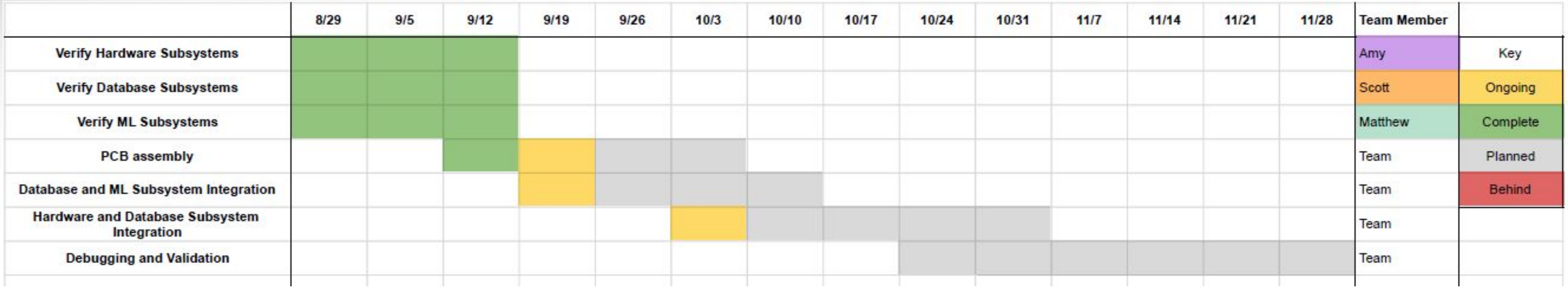
The Solution

- The goal of our project is to create a device with a secure caller ID/scam protection with a voice signature matching process that will reduce the ability of scammers to execute the 'grandma scam'.
- Our project will provide vocal signature matching to ensure that when family members or friends call they are recognized regardless of what number they call from.

Subsystems


Incoming call
Host computer
ESP32 board
- ring detection
- suppress ring
- hook flash detection
Voice recording
Power supply
Call information sent to database
Display caller ID information on OLED display
Display confidence interval on LED display
Amy Chen
Local storage for audio recording
Data processing
Database and cloud storage
Whitelist
Checked against NCID lists (whitelist/blacklist)
Blacklist
Stop call
Call recorded into database if not on either list
Scan for voice recognition
Scott Kevil-Yeager
Call passes
Scam or spam detection through machine learning
<75% confidence scam likely
Voice recognition through machine learning
>75% confidence scam likely
Matthew Hebrado

Project Timeline

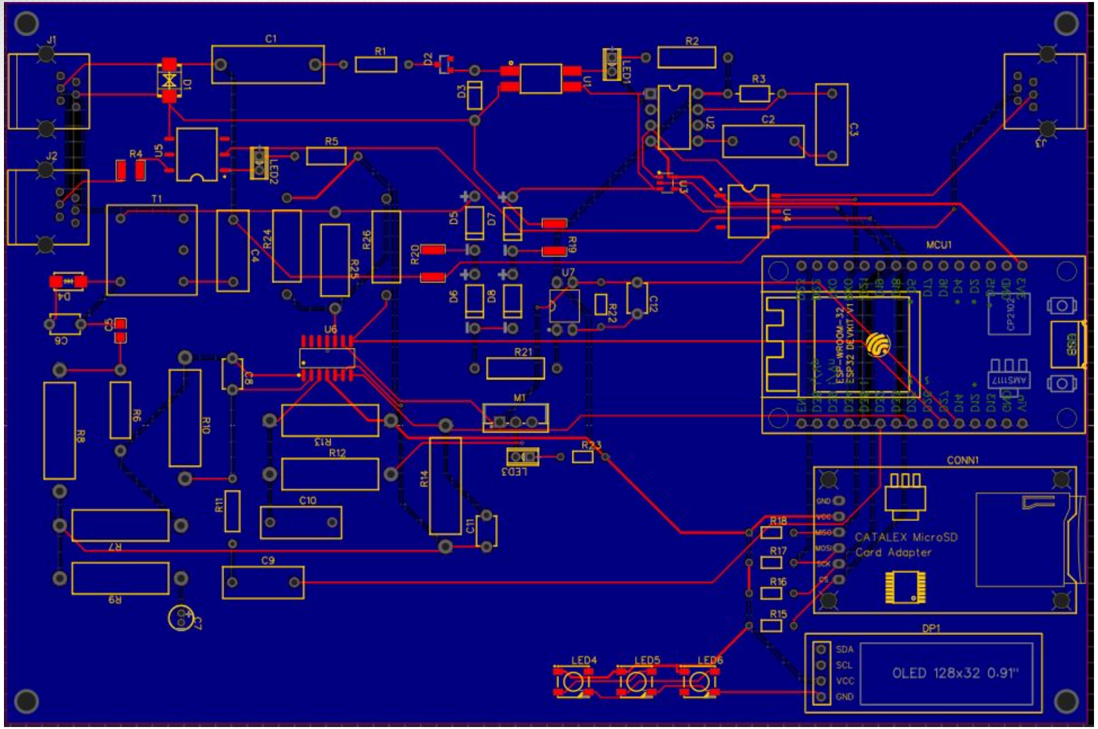


	8/29	9/5	9/12	9/19	9/26	10/3	10/10	10/17	10/24	10/31	11/7	11/14	11/21	11/28	Team Member	
Verify Hardware Subsystems	🟩	🟩	🟩												Amy	Key
Verify Database Subsystems	🟩	🟩	🟩												Scott	🟨 Ongoing
Verify ML Subsystems	🟩	🟩	🟩												Matthew	🟩 Complete
PCB assembly			🟩	🟨	▨	▨									Team	▨ Planned
Database and ML Subsystem Integration				🟨	▨	▨	▨								Team	🟥 Behind
Hardware and Database Subsystem Integration						🟨	▨	▨	▨	▨					Team	
Debugging and Validation									▨	▨	▨	▨	▨	▨	Team	

Hardware Subsystem

Accomplishments since last presentation 20 hrs of effort	Ongoing progress/problems and plans until next presentation
● Received PCB ● Created state machine diagram ● Began writing state machine code ● Soldered components to PCB ● Create a list of edits for 2nd round of PCB.	● Finish state machine code ● Test PCB ● Test state machine code ● Begin integrating with Scott

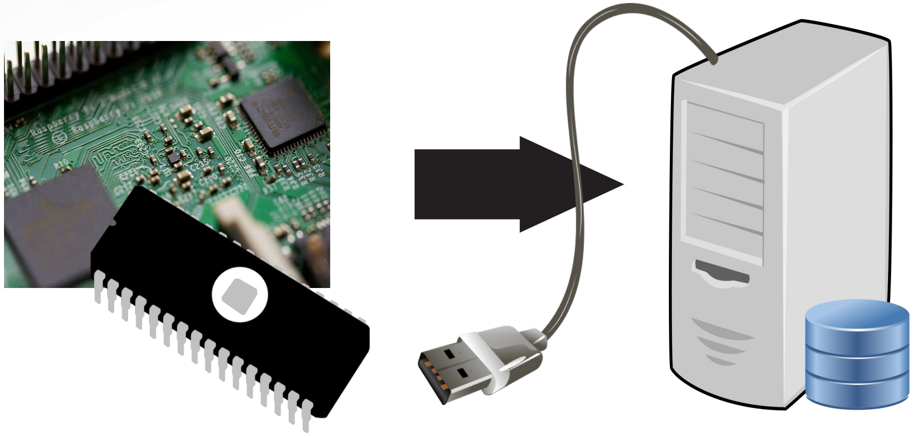
Hardware Subsystem



The hardware subsystem receives the incoming call signal from the landline phone and feeds the signal into the MCU (ESP32) for signal processing.

The ESP32 records calls and detects caller ID, caller waiting ID, and hook flash occurrences. The ESP32 will send data information to the host computer and database through JSON packages.

Database and Data Processing Subsystem Overview



The database and data processing subsystem acts as a bridge between the hardware and machine learning subsystems

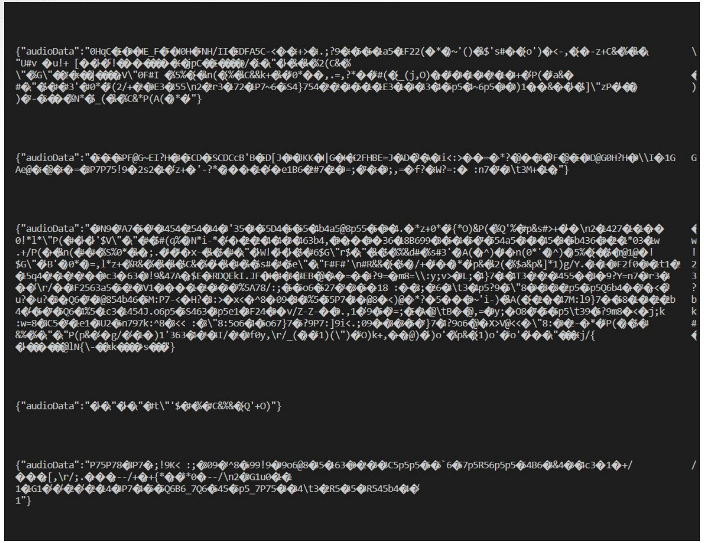
The ESP32 sends that data through serialized JSON packets to the host computer where it is decoded and audio data is sent to the database to be stored, and handset signals are parsed and converted into NCID’s proprietary gateway packeting system

Database and Data Processing

Accomplishments since previous presentation	Ongoing progress/problems and plans until next presentation
● Implemented sending serial packets of information, including variables for state machine information and audio data, from ESP32 to host computer (hardware -> database subsystem integration) ● Decoding json packet on host computer to prepare for it to be sent through the NCID platform using NCID gateway protocol (database -> NCID and machine learning subsystem integration) ● Testing minimum specs of host computer ● Helped solder components to PCB	● Add state machine code to audio recording and data transfer code in order to begin debugging any potential errors ● Help with PCB version A in order to have the quickest possible turn around ● Continue working on CID/CWID code with Amy (hardware - > database subsystem integration)

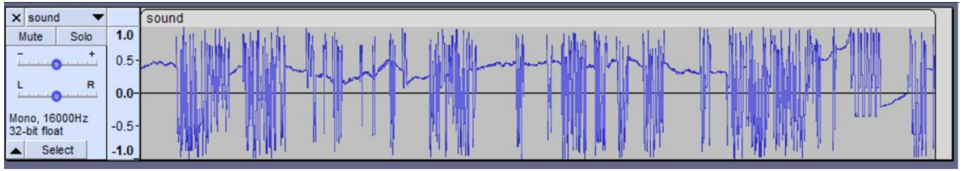
Database and Data Processing

Raw audio data in JSON packet:

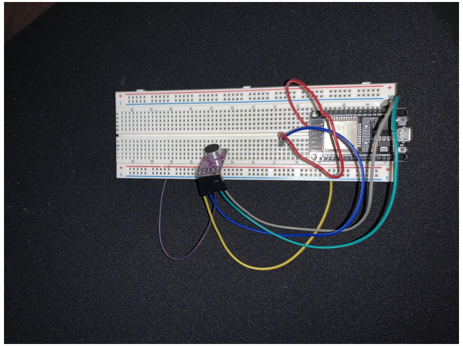


```
{"audioData":"0HqC�������NE_F���N0H��NH/II��DFA5C-<����+>��.;?9������a5��F22(�*�)~'()��$'s#����o')��<-,����-z+C&����   \
"U#v �u!+ [����F!�������������pC����������/���\"���&�&��2(C&�%
\"��G\"���������V\"0F#I ��5%���&n(���%�C&&k+��*0*��,.=,?*����#(��_(j,O)������������+��P(��a&�   �
#�\"����#3'�#0*��(2/+����E3��55\n2��r3��72��P7~6��S4}754������������E3����33��p5��~6p5����)1��������$]\"zP���)   )
)��-�����N*��_(���C&*P(A(�)*�"}

{"audioData":"�����PF@G~EI?H����CD���SCDCcB'B���D[J����KK��|G�����2FHBE=J���D��A���i<:>����=��*?����������F����D@G0H?H���\\I��1G   G
Ae@���@������=���P7P75!9��2s2���z+��'-?*���������e1B6��#7����=;������;,=��f?��W?=:�� :n7������\t3M+����"}

{"audioData":"��N9��A7������454��254��4��'35����5D4������4b4a5@8p55�������.�*z+0*��{*O)&P(���Q'%��p&s#>+����\n2���427������   �
0!*l*\"P(�������'$V\"���\"���$#(q%�)N*i-*�������������63b4,�������36���8B699����������554a5��������5�����b436������*03��w   w
.+/P(�����n(�������S%0*���;.��������x-���$�#��\"��W!������#6$G\"r$��\"���$����&d#��s#3'��A(���^)����n(0*'��^)��5%��������m@1@��!   !
$G\"��B'��0*��=,l*z+��R&�������$C&����������s#�����e\"��\"F#F#'\n#R&&�����)/+���������*��p&��2(���$a&p&]*1)g/Y.����0F2f0����t1��   2
��5q4��������c3��63��!9&47A��$E��RDQEkI.JF�����������BEB��@�A��=����?9=��m8=\\:y;v>��L;���}7����T3��������455����8���9?Y=n7���r3��   3
����\r/����F2563a5������V1���������������%5A78/:;����o6���27�����������18 :����;����26��\t3���p5?9���\"8������������p5���p5Q6b4����������<��   ?
u?��u?������Q6����@854b46���M:P7-<���H?���:>��x<���^8���09����%5���5P7������@8��<)@��*?���5����������~'i-)���A(�������47M:l9}7����5��������b   b
4���������Q6���%5���c3���454J.o6p5���S463���p5e1���F24����v/Z-Z-����.,1����=;�����@\tB��@,=���y;����O8���������p5\t39����?9m8���<��j;k   k
:w=8����C5����e1���U2���n797k:^8����<< :��\"8:5o6����o67}7��?9P7:]9i<.;09������������}7���?9o6�����X>V@<<��\"8:����2-���*��P(���$��#   #
&%����\"���\"P(p&�����g/�����)1'363������I/������f0y,\r/_(���*1)(\")��O)k+,������@)��)o'��p&��1)o'��o'����\"����j/{   �
���������@lN{\-�����k�������s����"}

{"audioData":"����\"����\"��t\"'$����%�#C&%&��(Q'+O)"}

{"audioData":"P75P78���P7��;!9K< :;���809���^8���99!9���9o6@8���5���63��������C5p5p5��������`6���7p5R56p5p5���4B6���&4�����c3��1��+/   /
�����[,\r/;.��������--/+���+{*�����*0��--/\n2���G1u0������
1���G1��������������4���P7��������Q6B6_7Q6���45���p5_7P75����4\t3���R5���5���R545b4�����
1"}
```

Audio data in audacity:



ESP32 bread boarded microphone:



Database and Data Processing

Future plans:

- Integration of database with ML algorithm; incoming and outgoing call voice prints stored to document variable to keep records of incoming and outgoing calls
- Integration with hardware audio recording through the ESP32 microcontroller allowing direct recording from handset, as well as a state machine for every hand set state (on-hook, off-hook, call waiting, etc.)
- Allow users to set a file directory, using a config file, that will hold recording files that will be automatically uploaded to the database, improve naming scheme of recordings to ensure easy look up and uploading

Additional plans:

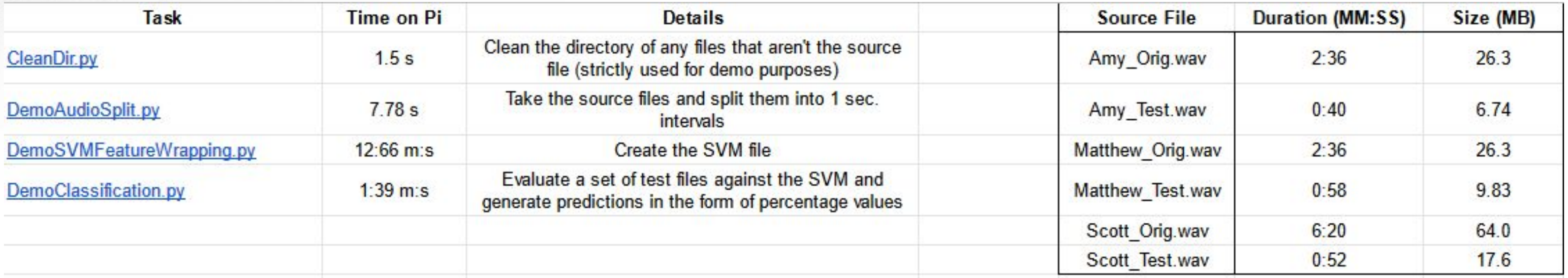
- Create NCID gateway test files for final release
- Continue commenting and documenting code for a README file at the end of the semester
- Continue working to clean up audio for machine learning subsystem

Machine Learning

Accomplishments since previous presentation 12 hrs of effort	Ongoing progress/problems and plans until next presentation
● I was able to successfully get my code running on my pi and timed how long it took the code to execute ● Included a filter that drops any voice reading with <75% match to any known voices to avoid false pairings	● Work on audio file management to prepare for integration with database subsystem ● Optimize the code to run faster on the pi

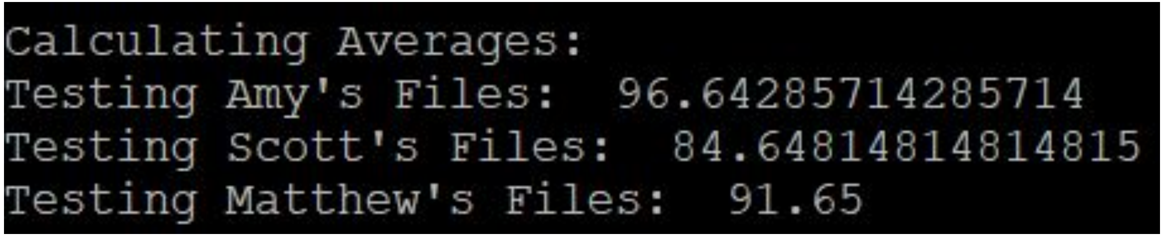
Machine Learning

Code Timings on Raspberry Pi



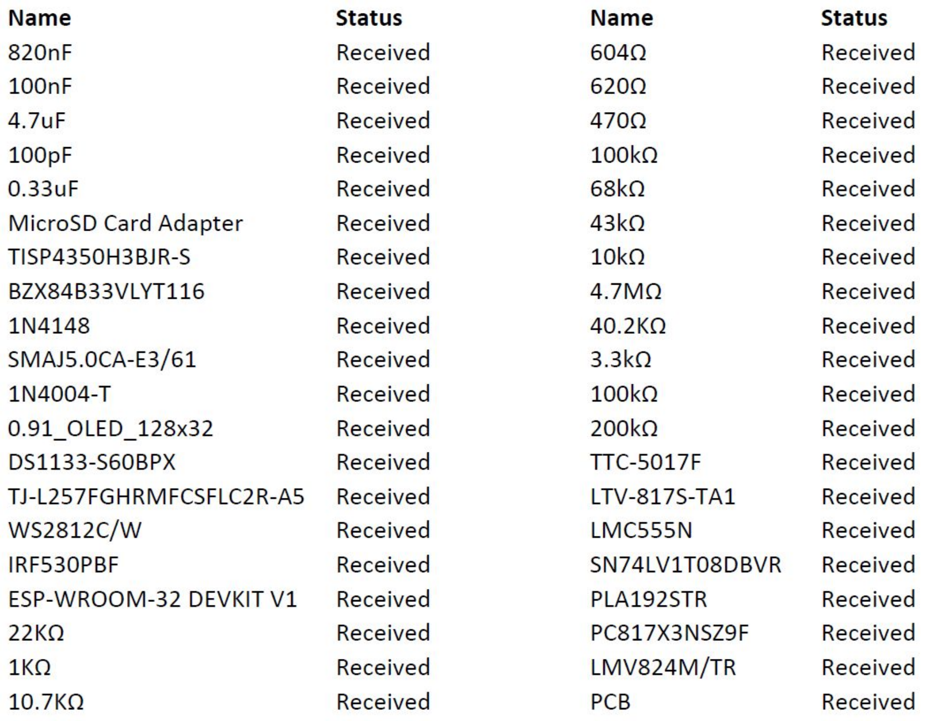
Task	Time on Pi	Details		Source File	Duration (MM:SS)	Size (MB)
CleanDir.py	1.5 s	Clean the directory of any files that aren't the source file (strictly used for demo purposes)		Amy_Orig.wav	2:36	26.3
DemoAudioSplit.py	7.78 s	Take the source files and split them into 1 sec. intervals		Amy_Test.wav	0:40	6.74
DemoSVMFeatureWrapping.py	12:66 m:s	Create the SVM file		Matthew_Orig.wav	2:36	26.3
DemoClassification.py	1:39 m:s	Evaluate a set of test files against the SVM and generate predictions in the form of percentage values		Matthew_Test.wav	0:58	9.83
				Scott_Orig.wav	6:20	64.0
				Scott_Test.wav	0:52	17.6

Output of DemoClassification.py



```
Calculating Averages:
Testing Amy's Files:  96.64285714285714
Testing Scott's Files:  84.64814814814815
Testing Matthew's Files:  91.65
```

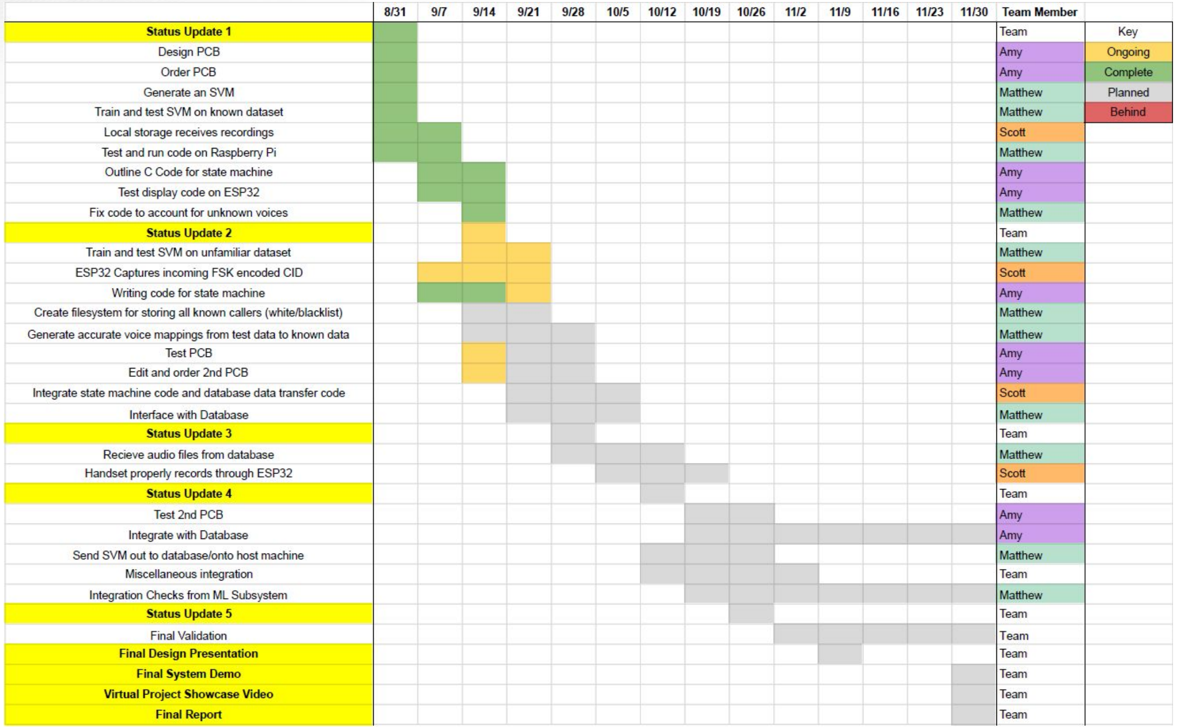
Parts Ordering Status



Name	Status	Name	Status
820nF	Received	604Ω	Received
100nF	Received	620Ω	Received
4.7uF	Received	470Ω	Received
100pF	Received	100kΩ	Received
0.33uF	Received	68kΩ	Received
MicroSD Card Adapter	Received	43kΩ	Received
TISP4350H3BJR-S	Received	10kΩ	Received
BZX84B33VLYT116	Received	4.7MΩ	Received
1N4148	Received	40.2KΩ	Received
SMAJ5.0CA-E3/61	Received	3.3kΩ	Received
1N4004-T	Received	100kΩ	Received
0.91_OLED_128x32	Received	200kΩ	Received
DS1133-S60BPX	Received	TTC-5017F	Received
TJ-L257FGHRMFCSFLC2R-A5	Received	LTV-817S-TA1	Received
WS2812C/W	Received	LMC555N	Received
IRF530PBF	Received	SN74LV1T08DBVR	Received
ESP-WROOM-32 DEVKIT V1	Received	PLA192STR	Received
22KΩ	Received	PC817X3NSZ9F	Received
1KΩ	Received	LMV824M/TR	Received
10.7KΩ	Received	PCB	Received

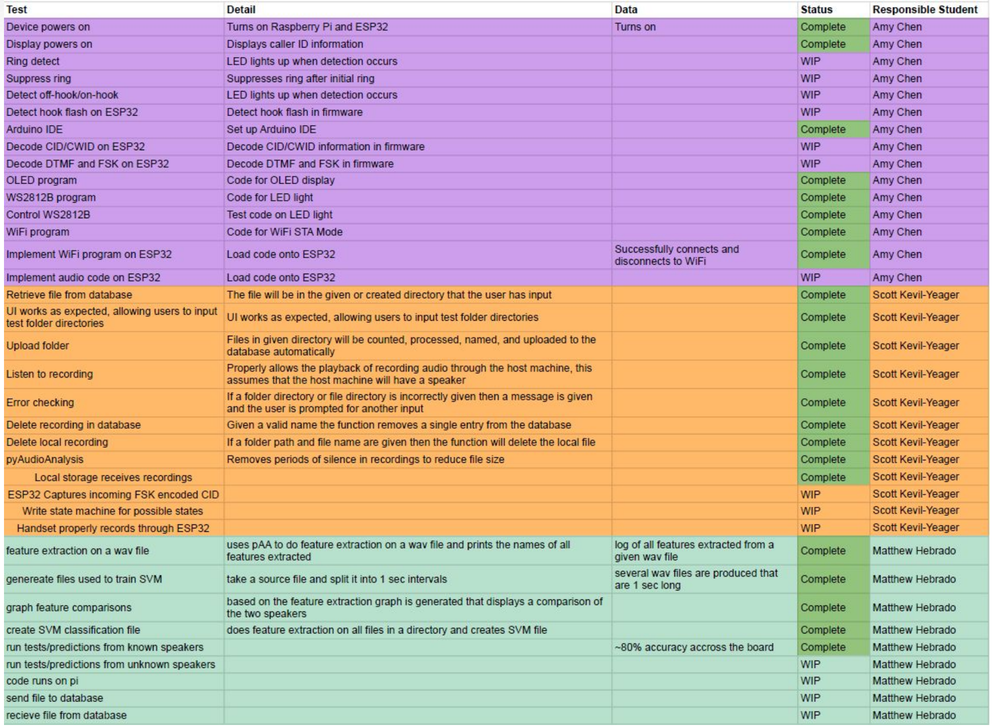
Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Execution Plan



	8/31	9/7	9/14	9/21	9/28	10/5	10/12	10/19	10/26	11/2	11/9	11/16	11/23	11/30	Team Member	
Status Update 1	🟩														Team	Key
Design PCB	🟩														Amy	🟨 Ongoing
Order PCB	🟩														Amy	🟩 Complete
Generate an SVM	🟩														Matthew	▨ Planned
Train and test SVM on known dataset	🟩														Matthew	🟥 Behind
Local storage receives recordings	🟩	🟩													Scott	
Test and run code on Raspberry Pi	🟩	🟩													Matthew	
Outline C Code for state machine		🟩	🟩												Amy	
Test display code on ESP32		🟩	🟩												Amy	
Fix code to account for unknown voices			🟩												Matthew	
Status Update 2			🟨												Team	
Train and test SVM on unfamiliar dataset			🟨	🟨											Matthew	
ESP32 Captures incoming FSK encoded CID		🟨	🟨	🟨											Scott	
Writing code for state machine		🟩	🟩	🟨											Amy	
Create filesystem for storing all known callers (white/blacklist)			▨	▨											Matthew	
Generate accurate voice mappings from test data to known data			▨	▨	▨										Matthew	
Test PCB			🟨	▨	▨										Amy	
Edit and order 2nd PCB			🟨	▨	▨										Amy	
Integrate state machine code and database data transfer code				▨	▨	▨									Scott	
Interface with Database				▨	▨	▨									Matthew	
Status Update 3					▨										Team	
Recieve audio files from database					▨	▨	▨								Matthew	
Handset properly records through ESP32						▨	▨	▨							Scott	
Status Update 4							▨								Team	
Test 2nd PCB								▨	▨						Amy	
Integrate with Database								▨	▨	▨	▨	▨	▨	▨	Amy	
Send SVM out to database/onto host machine							▨	▨	▨						Matthew	
Miscellaneous integration							▨	▨	▨	▨					Team	
Integration Checks from ML Subsystem								▨	▨	▨	▨	▨	▨	▨	Matthew	
Status Update 5									▨						Team	
Final Validation										▨	▨	▨	▨	▨	Team	
Final Design Presentation											▨				Team	
Final System Demo														▨	Team	
Virtual Project Showcase Video														▨	Team	
Final Report														▨	Team	

Validation Plan



Test	Detail	Data	Status	Responsible Student
Device powers on	Turns on Raspberry Pi and ESP32	Turns on	Complete	Amy Chen
Display powers on	Displays caller ID information		Complete	Amy Chen
Ring detect	LED lights up when detection occurs		WIP	Amy Chen
Suppress ring	Suppresses ring after initial ring		WIP	Amy Chen
Detect off-hook/on-hook	LED lights up when detection occurs		WIP	Amy Chen
Detect hook flash on ESP32	Detect hook flash in firmware		WIP	Amy Chen
Arduino IDE	Set up Arduino IDE		Complete	Amy Chen
Decode CID/CWID on ESP32	Decode CID/CWID information in firmware		WIP	Amy Chen
Decode DTMF and FSK on ESP32	Decode DTMF and FSK in firmware		WIP	Amy Chen
OLED program	Code for OLED display		Complete	Amy Chen
WS2812B program	Code for LED light		Complete	Amy Chen
Control WS2812B	Test code on LED light		Complete	Amy Chen
WiFi program	Code for WiFi STA Mode		Complete	Amy Chen
Implement WiFi program on ESP32	Load code onto ESP32	Successfully connects and disconnects to WiFi	Complete	Amy Chen
Implement audio code on ESP32	Load code onto ESP32		WIP	Amy Chen
Retrieve file from database	The file will be in the given or created directory that the user has input		Complete	Scott Kevil-Yeager
UI works as expected, allowing users to input test folder directories	UI works as expected, allowing users to input test folder directories		Complete	Scott Kevil-Yeager
Upload folder	Files in given directory will be counted, processed, named, and uploaded to the database automatically		Complete	Scott Kevil-Yeager
Listen to recording	Properly allows the playback of recording audio through the host machine, this assumes that the host machine will have a speaker		Complete	Scott Kevil-Yeager
Error checking	If a folder directory or file directory is incorrectly given then a message is given and the user is prompted for another input		Complete	Scott Kevil-Yeager
Delete recording in database	Given a valid name the function removes a single entry from the database		Complete	Scott Kevil-Yeager
Delete local recording	If a folder path and file name are given then the function will delete the local file		Complete	Scott Kevil-Yeager
pyAudioAnalysis	Removes periods of silence in recordings to reduce file size		Complete	Scott Kevil-Yeager
Local storage receives recordings			Complete	Scott Kevil-Yeager
ESP32 Captures incoming FSK encoded CID			WIP	Scott Kevil-Yeager
Write state machine for possible states			WIP	Scott Kevil-Yeager
Handset properly records through ESP32			WIP	Scott Kevil-Yeager
feature extraction on a wav file	uses pAA to do feature extraction on a wav file and prints the names of all features extracted	log of all features extracted from a given wav file	Complete	Matthew Hebrado
genereate files used to train SVM	take a source file and split it into 1 sec intervals	several wav files are produced that are 1 sec long	Complete	Matthew Hebrado
graph feature comparisons	based on the feature extraction graph is generated that displays a comparison of the two speakers		Complete	Matthew Hebrado
create SVM classification file	does feature extraction on all files in a directory and creates SVM file		Complete	Matthew Hebrado
run tests/predictions from known speakers		~80% accuracy accross the board	Complete	Matthew Hebrado
run tests/predictions from unknown speakers			WIP	Matthew Hebrado
code runs on pi			WIP	Matthew Hebrado
send file to database			WIP	Matthew Hebrado
recieve file from database			WIP	Matthew Hebrado

Thank You!