**GEMY V1.0.0**

**Documentation ….**

***IoT Training Project Eng. Mohammed Hatem***

***Team***:

* Taha Fawzy Anwar Elshrif
* Mazen ElSayed
* Omar Essam
* Omar Mohab

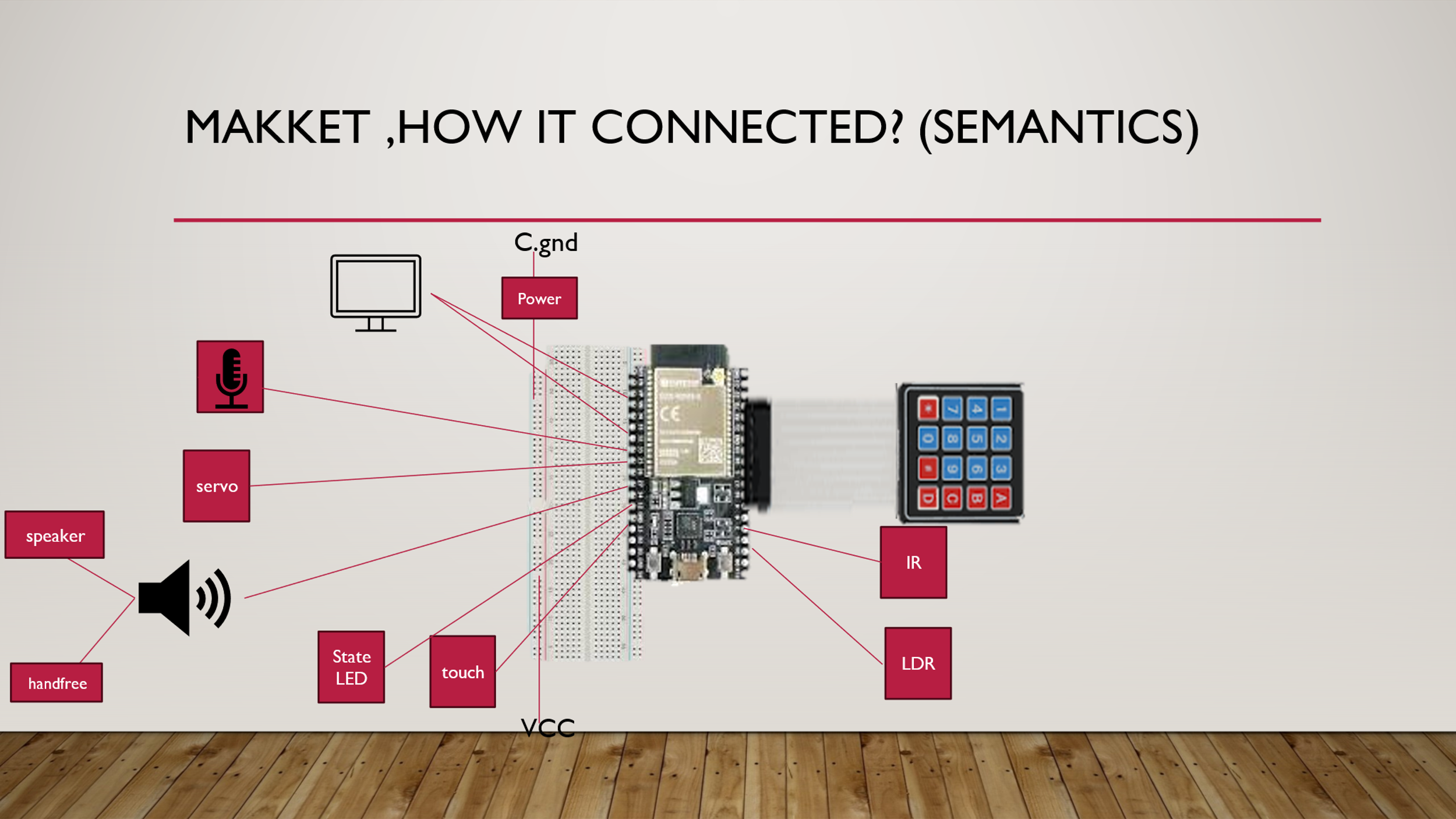
***What is this***

* *Gemy is simple Maquette simulate a robot that implements Gemini API to fetch results*
* *The main project idea is to be your friendly robot for chat, this version 1: implements chat using the mobile app and LCD output*

***Specifications****:*

* *LCD to display response output*
* *Android app to send message*
* *The app has feature to control the app – and get app logs (Firebase rule)*
* *Hello when near robot - light – touch hand*

***Semantics of connection:***



*Where:*

*Pins*

* *playing\_led\_pin=18 ##intialization ok or giving messages*
* *ldr\_pin=34*
* *ir\_pin=35*
* *touch\_pin = TouchPad(Pin(4))*
* *rows = [Pin(i, Pin.OUT) for i in (32,33,25,26)]#(13,12,14,27)]*
* *cols = [Pin(i, Pin.IN, Pin.PULL\_UP) for i in (27,14,12,13)]*
* *Sevo :15*
* *Mic\_pin=02*
* *Speaker :5*

*Spec.*

* *VCC are all on one line breadboard (+) (except LCD need VIN)*
* *Ground in the - line*
* *Power is just cutting the wire of the ground*
* *The state led is 2 led in parallels (with resistor)*
* *LDR is in simple PCB in cartoon*
* *The speaker and hand free port connected in parallel due to limitation of ports*

***Structure:***

*-In order to implement the chat, we need to call the Gemini API to send and receive message.*

*-Unfortunately: the python Gemini Api not supported for esp32 either micropython or C*

*- so, we need to use server module*

*We have 2 solutions:*

*1- Azure Function (the free function available right now)*

*2-Mqtt server*

*The first not efficient as it depends on http request so slow so won’t be the efficient user experience*

*-for me MQTT worked better in micropython ,so project is micropython*

***- mqtt server is just python script***

***-the API is*** *test.mosquitto.org as it free*

***1-Python server:***

*A screen shot of a computer

Description automatically generated*

*A screen shot of a computer

Description automatically generated*

* The UseGeminiAPI : take message and the apikey and call Gemini Api with the message
* For this version just have one topic for message ,responses for all users (,for later ISA will for each user for security)

***1-Micropython (ESP32 code):***

*-for better structuring : the code is divided into modules*

*Modules and main merhods:*

***Firebase*** *:*

* *Main methods for get and send to \from firebase*

*firebase\_authenticate: authenticate to firebase (first to call)*

*send\_data\_firebaseAuth()*

*get\_data\_firebaseAuth()*

*and get\_data\_firebase()… just helper (if no authentication ,not used)*

***nvs\_Store*** *:*

*-Main methods to deal with NVS (ESP32 permanent memory)*

*-what stored here? Is The firebase account – Wifi credentials to make not login each time for better UX*

*-* *esp\_NVS='gemy\_space'*

*storeStringStore() :store value (key-value pair)*

*getStringStore() :get value ,given key ,return “” if not found*

*clear\_keyStore() :clear key*

***Servo****:*

*-main methods to control the servo*

*-* *set\_servo\_angle() :for micropython ,it has limit that it don’t have library for servo like C ardinuo ,so this function just given angle : move servo*

*-* *helloServo() :function for moving when* ***hello******condition(I mean by hello when you near robot /give light /touch it say hello)***

*-* *optimized\_helloServo() :sometimes servo not worked so we call this instead hello ,idea of it is to move small angle each time*

*🡪 this is just core function to move servo in hello ,but the whole hello process is in utils module*

*-*

***Speaker****:*

*-main methods to control the speaker*

*-play\_tone() :play tone frequency with duration*

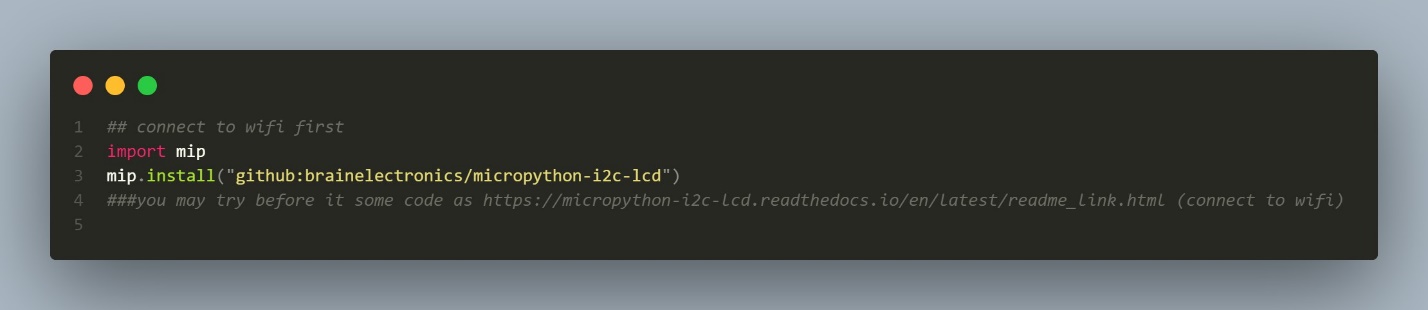
*playFreqTone() : play tone frequency ,given array*

*playHello() :play speaker for hello condition (the sound here is just random music)*

*default\_Mute(): sometimes when call speaker it give small noise ,so this method deinit the PWM*

***lcd\_eye :***

***-uses lcd\_i2c library ,to install :***



*-methods for displaying eye & others for LCD*

*-The eye is custom char displaying eye*

*-to make it large it’s implemented in more than char (8 char)*

*draw\_eye():given index start to draw eye given index*

*setupeye() :setup eye ,just lcd.create\_char(0, eye\_11) for each char*

*blink\_eyes\_one\_time(), blink\_eyes() :display eye ,blinking (each small time clear LCD for small time) ,the blink\_eyes\_one\_time is more used as main code is already implemented while True*

*prepareLcd():some times after boot LCD not prepared , so this method prepare it ,just call blink\_eyes\_one\_time two times*

***Utils*** *:*

*-implemented other utils ,no direct need to be in one class*

*-define ports ,and keypad matrix*

*-* *get\_text\_touchpad\_until(stop\_char) ,* *get\_text\_until\_touch :get text input until stop\_char ‘D’ used or touch (not used)*

*-* *connect\_wifi() :given ssid ,password connect to it ,the methods connect to wifi ,but actually instead to be while True ,try for time period ,if not make user reinput*

*-* *connect\_wifi\_input\_keypad(): connect to wifi but given input from keypad ,first it try to get from nvs\_store ,if not connectable ,input user ,the process (input->connect) is repeated until connect ,then store new in the NVS Store*

*-connect\_firebase\_input\_keypad\_store() :login ,given input from keypad ,like wifi : ,first it try to get from nvs\_store ,if not connectable ,input user ,the process (input->connect) is repeated until login ,then store new in the NVS Store*

*-* *read\_LDR() ,read\_IR() :to read IR ,ldr ,touch values*

*-check\_touched() :read touch pin ,and if small so touched so return true ,for long touch may give error so “try” to return true if long touch*

*-🡪 The thresoulds for touch ,ldr ,ir is selected from best tried*

*-hello\_condition() :for one time check hello condition ,if true play servo and speaker*

*-hello\_condition\_firebase ():like before for one time check hello condition ,if true play servo and speaker ,but before it ,it ensure that in firebase it’s enabled (Settings)*

*-* *repeated\_hello\_firebase():call the above method ,but repeated while True ,this is used in the thread*

***Mqtt\_prepare\_no\_mic:***

*-the first ,main module ,invoke main methods , use mqtt server*

*-* *mqtt\_reconnect():reconnect if disconnected*

*-* *mqtt\_callback :set callback to receive messages*

*-structured like C :setup ,loop*

*-Setup () : defaultmute() 🡪prepareLCD() 🡪* *connect\_wifi\_input\_keypad()🡪connect\_firebase\_input\_keypad\_store and save token in variable🡪* *blink\_eyes\_one\_time() (to give eye as started)🡪connect to mqtt 🡪send hello to server*

*-after setup :the app initialize thread for repeated\_hello\_firebase ,to make it happen concurrently beside the main loop (to not wait it)*

*-then main loop*

*--sometimes to save memory we call gc.collect() to collect garbage collection*

*-the output may be large if message is large ,so streamly message is loaded as 32char each time*

***Android App:***

***Utils*** *:*

*same idea like python : main helper methods for clean overall work*

*goToActivity() ,* *showDialog()*

*setEmailPassword() :after login load the account to memory , save info in shared preferences for later sessions*

*loadSharedPrefrence() :load it when opening*

*-endFirstTime (): if first time ,display guide ,if not display menu this is done by Boolean variable in shared preferences*

*-* *madeBlock() : for some activities (LOG , FAQ) : each question /log is a block added to view ,the instance is created from this method*

*createDefaultKeys() :when create account :create other variables (node childs)*

*authenticateAccount() : login (invoked also in login after createAccount)*

***MQTTHandler****:*

*-this is class to easy use MQTT*

*-the app send message to 'gemy/user/messages' ,and output received in 'gemy/user/responses'*

***MainActivity****:*

*-this is the splash screen*

***Guide1****:*

*-this is the first guide ,tell you to connect to wifi,in valid range chars (A-C \* # 1-9 ,D to send) ,guide you to hotspot*

***Guide2 /new user /log in****:*

*-this is the second guide ,tell you to create user /log in in valid range chars*

*- currently authentication is by email + password to firebase ,email is* [*username+@gemy.com*](mailto:username+@gemy.com)

***Faq :***

* *Frequently asked questions ,just made many Utils blocks for some questions*

***Logs****:*

* *Error /state from robot*

*This by : any change in firebase “error” send new Utils block (currently not work ,logs this version :by specific messages send to server “not tested”)*

***About :***

*-about app 😊*

***Processing***

***1-****Stream processing:*

*Due to limit of esp32 memory ,it won’t store the whole response ,also MQTT can not exchange the big response*

*So to solve each time it loads 32 char (as 16\*2 screen) until end*

*2-Firebase:*

*-the firebase rule to be more secure :*

A screenshot of a computer

Description automatically generated

*And example of it:*

A screenshot of a computer

Description automatically generated

*Where :123456789 ,r is just users*

*Clear\_pass :if requested from app to logout*

*Clear\_session :if requested from app to clear wifi*

*Error :If error send logcat to app*

*Record : is recording?*

*helloCond :if requested from app to stop hello*

*speakerCond :if requested from app to stop speaker in hello*

*language: language to spoken*

*-currently some settings not work as memoryFULL error*

*3- Secure :*

*-secure login to account (as rules :each user access his data only )*

*-for this version one limit :that it stores user password*

*Which must not be done*

*3- Concurrent :*

*Applying settings is done concurrently using firebase Realtime database*

*4-Thread processing:*

*The app operates in two threads : one for robot ,other for hello*

*4- memory optimized :*

*-To save memory auth\_token (auth. To firebase ) ,quickly after generated it save to nvs\_store*

*-every time and time gc.collect() to clear garbage collection to save memory*

*-optimize imports*

*-stream processing*

***Limitations of Version :***

*- not totally secure in mqtt server*

*-limit in keypad*

*-although memory optimized it many times sometimes may produce memory leaks*

*-the code is optimized however ,for firebase code is work but in project give memory fill error ,that can not be solved right now (Send to server : worked in sending hello message ,and error/reconnect but not tested)*

*- for this version the way to receive message is not best one : that always looks as at public variable message ,and modify it from outside class*

*GitHub:*

*-* [*https://github.com/TahaFawzyElshrif/GEMY*](https://github.com/TahaFawzyElshrif/GEMY)



*Demo:*

[*https://drive.google.com/file/d/1OoTq3I9fxtxAOTANQALYn\_WZHaztmRLm/view?usp=sharing*](https://drive.google.com/file/d/1OoTq3I9fxtxAOTANQALYn_WZHaztmRLm/view?usp=sharing)

[*https://drive.google.com/file/d/16m1N2Ex4vi4AC1swpYj8I6dGdjysm521/view?usp=sharing*](https://drive.google.com/file/d/16m1N2Ex4vi4AC1swpYj8I6dGdjysm521/view?usp=sharing)

[*https://drive.google.com/file/d/1qvsTQOxUyhHvkRBUiXIPEyWHKeiiNKcG/view?usp=sharing*](https://drive.google.com/file/d/1qvsTQOxUyhHvkRBUiXIPEyWHKeiiNKcG/view?usp=sharing)

*LinkedIn:*

[www.linkedin.com/in/taha-elshrif](http://www.linkedin.com/in/taha-elshrif)