Disseny d’un sistema de control per a un pàncrees artificial comandat per Telegram

Daniel Silles Vizuete

Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació

Arduino

Antoni Morell Pérez

Pere Tuset Peiró

Gener 2020

FITXA DEL TREBALL FINAL

Títol del treball:	*Disseny d'un sistema de control per a un pàncrees artificial comandat per Telegram*
Nom de l'autor:	*Daniel Silles Vizuete*
Nom del consultor/a:	*Antoni Morell Pérez*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2020*
Titulació o programa:	*Grau en Enginyeria de Tecnologies i Serveis de Telecomunicació*
Àrea del Treball Final:	*Arduino*
Idioma del treball:	*Català*
Paraules clau	*Pàncrees, Telegram, Bot*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

El principal objectiu d'aquest treball és el disseny d'un sistema de control per a un pàncrees artificial, que estarà comandat mitjançant el programa de missatgeria de codi obert Telegram.

En la primera part d'aquest projecte es realitzarà el disseny i el muntatge d'un prototip de pàncrees artificial molt basic i que funcionarà de manera autònoma fent servir una placa Arduino UNO com a mòdul de control. Aquest prototip comptarà amb un display LCD on es visualitzarà la informació principal.

Un cop realitzat el primer prototip es realitzarà el muntatge i disseny del segon prototip basat en el primer però molt més avançat. D'entrada s'escollirà una placa de desenvolupament basada en el SoC ESP32 que afegirà connectivitat WiFi, millor processament i més memòria al sistema.

En primer lloc serà necessària la creació d'un Bot de Telegram que serà l'encarregat de realitzar les comunicacions amb el sistema. A més, s'hauran d'afegir unes rutines de seguretat i d'identificació en l'accés al Bot per tal de poder gestionar la seguretat en el sistema.

Finalment es comptarà amb un sistema d'alerta que permetrà al pacient viure amb més tranquil·litat, ja que en cas d'uns nivells no desitjats de glucosa s'enviarà un missatge

d’alerta a les persones que han de tenir cura del pacient.

Abstract (in English, 250 words or less):

This Project main aim is to design a control system for an artificial pancreas, which will be commanded by the open-source messenger platform Telegram.

On this project first part, the design and assembly of a very basic autonomous artificial pancreas prototype will be performed using an Arduino UNO board as a control module. This prototype will have an LCD display where the main information will be displayed.

Once the first prototype has been developed the assembly and design of the second prototype based on the first, but much more advanced, will be performed. An ESP32 SoC development-based-board will be chosen as main part, which will add WiFi connectivity, better processing and more memory to the system.

First, it will be necessary to create a Telegram Bot that will bring us the communication way with the system. In addition, security routines and access to the Bot must be added in order to be able to manage security of the system.

Finally, there will be an alert system that will allow the patient to live more calmly, in case of undesirable glucose levels an alert message to the patient’s carer will be sent automatically.

Índex de continguts

Índex de il·lustracions

Índex de esquemes

Índex de taules

1 Introducció

En la actualitat estem vivint a un moment on els dispositius basats en el IoT (*Internet of Things*) reclamen un lloc destacat a les nostres vides i aquests, cada cop més, omplen més aspectes quotidians de forma silenciosa.

L'alt nivell de estrès, la contaminació i els hàbits de vida poc saludables, afavoreixen a l'apareixement d'algunes malalties cròniques, per sort, la ciència i la medicina cada dia avancen més ràpid i poc a poc aquestes que en un passat eren mortals, avui en dia es poden suportar d'una forma força més lleu gracies als avenços tecnològics.

En aquest treball es parlarà de la diabetis i dels dispositius que afavoreixen una millora en el tractament d'aquesta malaltia, els anomenats pàncrees artificials, i es pretendrà fer una millora en les prestacions que ofereixen apropant-los una mica més al IoT.

Així doncs, sembla interesant dissenyar noves vies d'accés als dispositius mèdics amb noves rutines amb les quals els usuaris ja estan prou familiaritzats com pot ser un simple programa de missatgeria. I a més l'adició de connectivitat a Internet pot obrir una porta a noves i millors línies d'investigació.

2 Objectius del projecte

La finalitat d'aquest projecte es conèixer i treballar amb plaques basades en Arduino i explorar la capacitat d'adaptació del codi, el control de perifèrics i les comunicacions a traves de llibreries molt senzilles d'utilitzar de tot el ecosistema d'aquest tipus de plaques.

També es pretén realitzar un petit estudi d'algunes de les tecnologies emprades en aquestes plaques per realitzar les comunicacions amb els perifèrics i visualitzar fàcilment com la tria de certs components o sistemes de comunicació ens simplifica substancialment el circuït de manera física.

Es pretén el disseny de un dispositiu basic amb la capacitat d'emular algunes característiques principals d'un pàncrees artificial. Els prototips disposaran d'un codi senzill que facilitarà l'adició i la escalabilitat del codi cap a nous prototips més avançats i sense problemes per afegir noves característiques i el control de nous perifèrics amb l'ajuda de les corresponents llibreries.

Es faran servir patrons i dades estàndards, doncs els paràmetres basals i el FSI (Factor de Sensibilitat a la Insulina) entre d'altres han d'estar establerts convenientment per un endocrí a traves d'un seguiment temporal exhaustiu.

Es donarà èmfasi en la importància del control del dispositiu fent servir una aplicació de missatgeria d'una forma clara i entenedora i la importància dels avisos a través d'aquests tipus d'eines, ja que obren la porta a un millor control d'aquestes malalties en pacients dependents per part de les seves famílies.

3 Motivacions

Una de les principals motivacions d'aquest projecte és poder aprofundir en els dispositius que fan més fàcil el dia a dia als pacients d'una malaltia crònica, com és el cas de la diabetis, i millorar algunes de les mancances en les prestacions que els anomenats pàncrees artificials ofereixen.

En el meu cas, he viscut de molt a prop la diabetis, la meva germana va ser diagnosticada des de molt petita i he pogut observar la evolució en els tractaments de la malaltia.

Si bé és cert, que per a una persona que hagi conviscut molt de temps amb la malaltia i hagi rebut una formació adequada en el tractament no son necessàries grans explicacions o funcionalitats, existeixen els casos de persones dependents que potser requereixen d'un seguiment exhaustiu per alguna altra persona responsable.

És per aquest motiu que em sembla una bona idea integrar els dispositius mèdics de seguiment en el IoT i en un futur poder recollir els resultats de forma anònima en una gran base de dades basada en *Open Data* per afavorir noves investigacions sobre la

malaltia, ja que aquestes propietats no venen contemplades en les prestacions dels models comercials.

4 Descripció del projecte

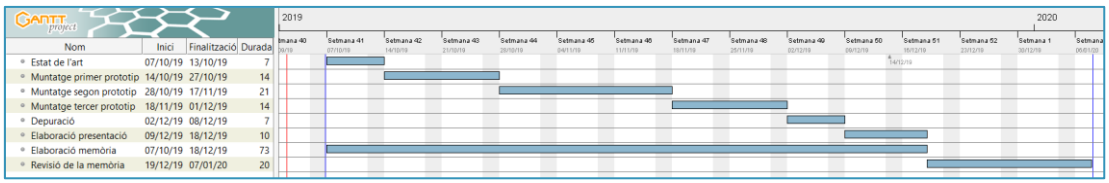
En aquest projecte, d'inici es realitzarà el muntatge de dos prototips, el primer es desenvoluparà mitjançant una placa Arduino UNO com a base i el segon amb una placa de desenvolupament ESP32, on es pretén simular un pàncrees artificial basic que permeti una comunicació via un programa de missatgeria.

- El primer prototip, el més simple, serà una versió independent on trobarem la informació que simularem i visualitzarem directament en un display i tindrem un actuador que emularà una bomba d'insulina. L'emulació de la velocitat de la bomba es realitzarà mitjançant un LED amb intensitat variable amb PWM.
- El segon prototip naixerà del primer, en aquest cas amb la placa de desenvolupament del ESP32 i gaudirà de connexió WIFI. S'afegirà la funcionalitat de poder controlar els sensors directament amb un bot de Telegram, i a més, afegirem una pantalla OLED connectada per I2C.

5 Planificació del treball (diagrama de Gantt)

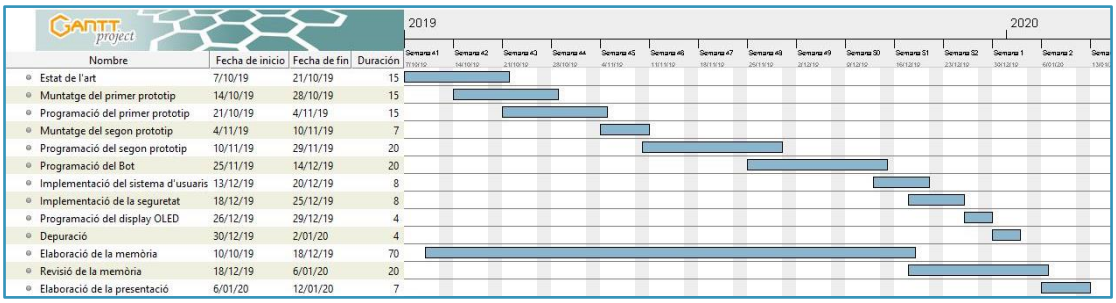
Es realitza una planificació del projecte basat, en un principi, en les dades d'organització de l'assignatura i es fa una estimació inicial de les tasques bàsiques que s'hauran de dur a terme.

A continuació es pot observar el diagrama de Gantt amb la organització de les tasques al principi del projecte:



TAULA 1. DIAGRAMA DE GANTT INICIAL

Un cop finalitzat el projecte es pot observar la diferencia amb les tasques previstes al inici del projecte:



TAULA 2. DIAGRAMA DE GANTT FINAL

6 Viabilitat del projecte

Ens trobem en un projecte que té una viabilitat difícil de definir. D'entrada, es pot definir fàcilment el preu dels prototips.

Preu del primer prototip:

Element	*Preu*
Placa Arduino UNO	20 €
Display LCD	5 €
Components electrònics	10 €
TOTAL	35 €

TAULA 3. PREU PRIMER PROTOTIP

Preu del segon prototip:

Element	*Preu*
ESP32 - DevKit	10 €
Display OLED	10 €
Components electrònics	10 €
TOTAL	30 €

TAULA 4. PREU SEGON PROTOTIP

A nivell pràctic obviarem el primer prototip, doncs ha sigut desenvolupat amb una finalitat bàsicament educativa i no té cap rellevància en cap futur projecte ni s'ha de tenir en compte amb cap propòsit.

Un cop definit els preu dels prototip, es poden revisar les possibles vies d'explotació del projecte. Per una banda, el podríem considerar com un producte stand-alone que complementaria un sistema de pàncrees artificial. En aquest supòsit seria necessari

prendre contacte amb els fabricants dels sensors i les bombes per tal de fer una col·laboració per actualitzar els seus productes i disposar de les llibreries de control d'aquests.

Per altra banda, es pot considerar el producte com un servei, que no només complementés un pàncrees artificial, si no, un servei que oferiria assistència i atenció a l'usuari. Això requeriria de la creació d'una infraestructura de persones o integrar el servei en alguna ja existent.

Com a ultima opció, per mi la desitjada, crear una plataforma oberta col·laborativa per desenvolupar un producte DIY (*Do It Yourself*) on qualsevol pogués dissenyar el seu pàncrees artificial amb una mínima inversió.

7 Descripció dels principals elements

7.1 La diabetis

La Organització Mundial de la Salut (OMS) defineix la diabetis com:

"*Una malaltia crònica que apareix quan el pàncrees no produeix insulina suficient o quan l'organisme no utilitza eficientment la insulina que produeix. L'efecte de la diabetis no controlada es la hiperglucèmia (augment del sucre en sang)*".

Segons el tipus d'afectació del pacient podem classificar la malaltia en tres variants:

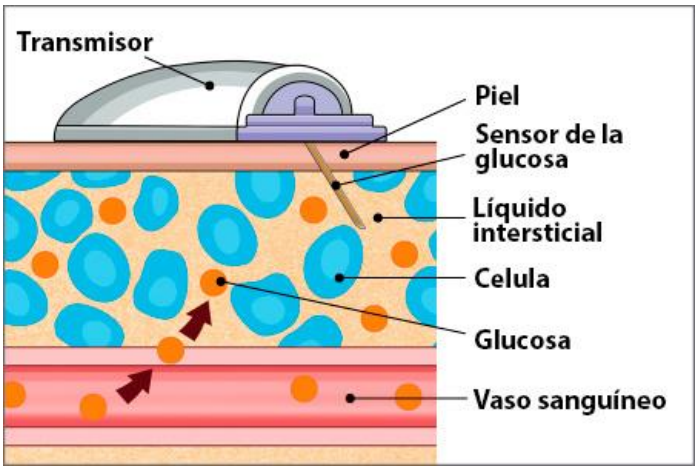
- La diabetis de tipus 1, que es caracteritza principalment per l'absència de síntesis de insulina.
- Diabetis tipus 2, es dona quan el cos no té la capacitat d'utilitzar d'una manera eficient la insulina. Sovint ve generada per un excés de pes o una activitat física deficient.
- La diabetis gestacional, apareix per primer cop durant l'embaràs i sol desaparèixer amb la finalització d'aquest. Les afectades per aquesta modalitat son susceptibles a desenvolupar la diabetis tipus 2 en un futur.

Existeixen diversos tractaments per la diabetis tipus 2 que no requereixen l'administració constant de insulina i inclús uns hàbits de vida més saludables poden millorar substancialment el tractament de la malaltia.

En canvi l'únic tractament eficient per a la diabetis tipus 1 es l'administració diària d'insulina ja que l'organisme no la produeix. I a més, es requereix un seguiment regular dels nivells de glucosa per part del pacient per tal de poder ajustar la dosi.

7.2 El glucòmetre

El glucòmetre es un dispositiu encarregat de mesurar la concentració de glucosa en sang d'un pacient. Existeixen dos tipus de mesures convencionals, el nivell de glucosa dels teixits (intersticial) i el nivell de glucosa en sang. Aquestes mesures no necessàriament comparteixen sempre el mateix valor, per exemple, el nivell de glucosa en sang fluctua més ràpidament. És per això, que les hem de considerar dos tipus de mesura diferents.



IL·LUSTRACIÓ 1. ESQUEMA DEL SENSOR DE GLUCOSA

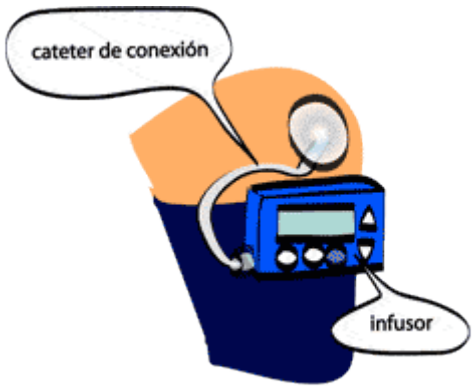
Els antics glucòmetres, funcionaven amb unes tires reactives que mitjançant l'encima *glucosa oxidasa*, provoca la oxidació de la glucosa, el que genera un canvi de color en el reactiu depenent de la quantitat de glucosa. Fent servir un procés de fotometria de reflectància es determina el nivell de glucosa. En canvi, actualment, en comptes de fer servir la fotometria, els mesuradors es serveixen de la tecnologia electroquímica per obtenir el nivell de glucosa, fent servir el petit corrent que es genera en la oxidació de la glucosa al reactiu.

7.3 La bomba de insulina

La bomba de insulina, és un petit dispositiu portàtil encarregat del subministrament de insulina de forma continuada durant tot el dia. Aquest dispositiu consta de dues parts principals: el infusor de insulina que és la part més complexa i el catèter de connexió per l'administració de la insulina.

- El infusor de insulina, que seria l'encarregat de administrar insulina de forma continua durant les 24 hores del dia. Aquest estaria format bàsicament per:
    - Un display
    - Un sistema de control
    - Un sistema d'alimentació
    - Un dipòsit de insulina
- El catèter de connexió, encarregat de fer arribar la insulina al teixit subcutani del pacient.

El control de la glucosa mitjançant la bomba de insulina té les seves particularitats, mentre que en l'administració per part del pacient de diverses dosis durant el dia es poden administrar diferents tipus de insulina depenent de la seva corba de resposta i les necessitats puntual, amb les bombes de perfusió continua de insulina, a llarg termini, s'adquireix un control més acurat.



IL·LUSTRACIÓ 2. PARTS PRINCIPALS DE LA BOMBA DE INSULINA

Aquests dispositius administren de forma continua una petita quantitat de insulina prefixada, que anomenarem insulina basal. Una quantitat de insulina basal correctament

dosificada, afegeix una menor variabilitat en la glucèmia del pacient durant el dia, a més, aquesta ha de ser convenientment adaptada a les principals rutines del pacient.

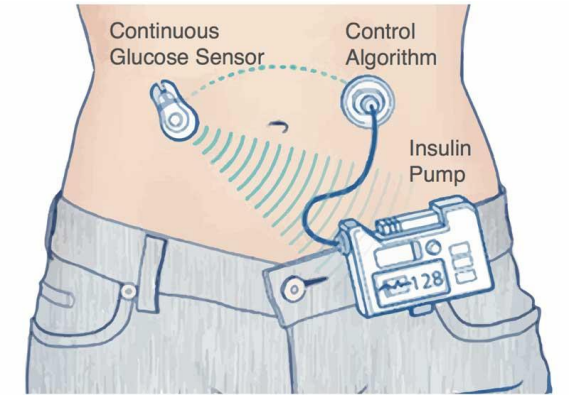
Puntualment en els moments de les ingestions s'han de preveure una sèrie de bolus, és a dir, administrar una quantitat extra de insulina per tal de corregir un nivell de glucosa elevat. S'ha de tenir en compte que la quantitat de insulina administrada en els bolus hauria de ser aproximadament la meitat al total de la insulina basal per norma general.

La bomba de insulina ofereix moltes millores en la qualitat de vida dels pacients diabètics es redueixen considerablement el nombre de punxades i dona un millor control de la diabetis, encara que per contra, és un dispositiu que ha d'estar en continu funcionament i requereix d'un manteniment periòdic.

7.4 El pàncrees artificial

Actualment la tecnologia va més enllà del simple control de glucosa i l'administració de la insulina controlada per el pacient. La principal línia de treball en el tractament de la diabetis tipus 1 i casos de tipus 2 mes greus, és el disseny del pàncrees artificial per tal de minimitzar els inconvenients de la malaltia i millorar les condicions de vida dels pacients.

Es coneix com pàncrees artificial, també anomenat sistema de nansa tancada per al control de la glucèmia o sistema d'infusió automàtica d'insulina, a aquell sistema electromecànic, no biològic, capaç d'infondre insulina de forma automàtica i dependent de la glucosa, emulant així la funcionalitat de la cèl·lula β pancreàtica, amb l'objectiu de mantenir la glucèmia en rangs normals i estables.



IL·LUSTRACIÓ 3. ESQUEMA DELS COMPONENTS D'UN PÀNCREES ARTIFICIAL

Bàsicament, el concepte de pàncrees artificial consisteix en tres parts principals:

- El sensor de glucosa, que determina el nivell de glucosa del pacient realitzant anàlisis constants.
- La bomba d'insulina, és un dispositiu que administra insulina al pacient a traves de una unitat de control, pot operar de forma manual o automàtica.
- La unitat de control, és la encarregada de rebre les dades del sensor de glucosa i fer d'actuador de la bomba d'insulina mitjançant uns complexos algoritmes, fer un monitoratge i controlar les comunicacions.

Malgrat els grans avenços en aquest tipus de dispositius en els últims anys, no existeix un sistema totalment automatitzat en el control de la glucosa, sinó que hem de referir-nos a sistemes híbrids que requereixen la intervenció del pacient en el moment dels àpats.

Això es deu a que la variabilitat dels nivells de glucosa i l'acció de la insulina no interaccionen de manera instantània, sinó que es serveixen d'unes corbes de resposta que sense l'acció del pacient són molt difícils de preveure.

Aquests dispositius, a més, es poden connectar a una base de dades externa on el pacient i el metge poden controlar la evolució de la malaltia d'una forma continua i visualitzar les dades i configuracions d'una forma senzilla des d'un dispositiu mòbil o un ordinador.

7.5 Arduino

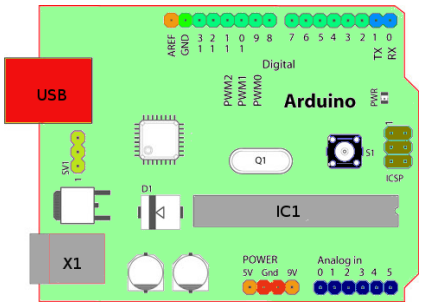
Per la banda de Arduino, es tracta d'un dels tipus de les plaques més populars del món *maker*, però que a diferència de la popular *Raspberry Pi* no compta amb un nombre limitat de models, sinó que ofereix unes bases de maquinari obert perquè altres fabricants puguin crear les seves pròpies plaques amb les seves variants.

El projecte va néixer el 2003, quan diversos estudiants de l'Institut de Disseny Interactiu de Ivrea, Itàlia, per tal de facilitar l'accés i l'ús de l'electrònica i programació a tothom. Van fer-ho perquè els estudiants d'electrònica tinguessin una alternativa més econòmica a les populars BASIC Stamp, unes plaques que aleshores valien més de cent dòlars, i que no tots es podien permetre.

Com a curiositat Arduino prové del nom del bar Bar di Re Arduino on Massimo Banzi (el principal creador d'aquestes plaques) passava algunes hores, el qual al seu torn ve del nom d'un antic rei europeu allà per l'any 1002.

Arduino és una plataforma de creació d'electrònica de codi obert, la qual està basada en maquinari i programari lliure amb llicencia *copyleft*, flexible i fàcil d'utilitzar per als creadors i desenvolupadors. Aquesta plataforma permet crear diferents tipus de microordinadors d'una sola placa als quals la comunitat de creadors pot donar-los diferents tipus d'ús.

A la pagina https://www.arduino.cc/en/Main/Products podem trobar totes les plaques oficials disponibles de marca *Genuino* (fora dels Estats Units) o *Arduino* (als Estats Units) encara que al tractar-se d'un projecte obert trobarem multitud de fabricants que produeixen plaques amb exactament les mateixes característiques i tecnologies i d'altres amb noves funcionalitats.



IL·LUSTRACIÓ 4. ESQUEMA DE LES PRINCIPALS CONNEXIONS DE ARDUINO

El maquinari consisteix en dissenys simples de maquinari lliure amb processadors *Atmel AVR* en una placa amb pins E/S. L'entorn de desenvolupament implementa el llenguatge *Processing* de *Wiring*, molt semblant a C++. Arduino es pot utilitzar per desenvolupar objectes interactius autònoms o pot ser connectat al programari d'un ordinador. Les plaques es poden muntar a mà o adquirir-se i els IDE de font oberta es poden descarregar de franc.

A més, és possible comunicar una aplicació que corri sobre Arduino amb altres dispositius que corrin altres llenguatges de programació i aplicacions populars, pel fet que Arduino fa servir la transmissió de dades en sèrie, la qual és suportada per la majoria dels llenguatges com per exemple Python.

7.6 Telegram

Telegram, és una aplicació de missatgeria instantània gratuïta i feta amb programari lliure que permet enviar i rebre missatges a través d'Internet. Permet crear grups i enviar imatges i vídeos. Un dels seus objectius és proveir una major privadesa i seguretat en comparació amb altres aplicacions similars. Ha estat desenvolupat pels germans Nikolai i Pavel Durov, els creadors de la xarxa social VK.

Amb cent milions d'usuaris, actualment és el tercer client de missatgeria mòbil pel que fa a nombre d'usuaris al món. Telegram és una aplicació de programari lliure, que pot ser modificada per la comunitat, pel que existeixen clients oficials per als sistemes operatius mòbils *Android* i *iOS*, i també per a ordinadors, així com versions per executar en el navegador.

Telegram té les següents característiques:

- Tots els missatges són encriptats i hi ha la possibilitat d'autodestrucció al cap d'un temps determinat de missatges i xats secrets.

- Permet enviar i rebre tota mena d'arxius i documents.

- Està basada en el núvol, per tant es pot accedir als missatges des de qualsevol dispositiu.

- Té un servei molt ràpid donat per la distribució de servidors escampats per tot el món.

- És una aplicació parcialment de codi lliure i gratuïta, sense anuncis.

- No té límit quant a la quantitat de dades que es poden enviar.

- El codi de costat de client (el programa que s'instal·la al telèfon mòbil) és de tipus codi obert, en canvi el codi de costat de servidor (programa dels ordinadors que ofereixen el servei Telegram) no és codi obert.

- Suporta la creació de bots.

Els bots són aplicacions de tercers que s'executen a l'interior de Telegram. Els usuaris poden interactuar amb els bots enviant-los missatges, ordres i peticions en línia. Els bots es controlen mitjançant les sol·licituds HTTPS a traves de la API bot de Telegram. Les principals tasques que els bots poden dur a terme son les següents:

- Obtenir notificacions i notícies personalitzades. Un bot pot actuar com un diari intel·ligent, i enviarà contingut rellevant tan aviat com es publiqui.

- Integrar-se amb altres serveis. Un bot pot enriquir xats de Telegram amb contingut de serveis externs.

- Acceptar els pagaments dels usuaris de Telegram. Un bot pot oferir serveis de pagament o treballar com a botiga virtual.

- Crear eines personalitzades. Un bot pot proporcionar alertes, previsions meteorològiques, traduccions, formats o altres serveis.

- Construir jocs. Un bot pot oferir experiències riques d'HTML5, des de jocs senzills i trencaclosques fins a jocs 3D i jocs d'estratègia en temps real.

- Crear serveis socials. Un bot pot connectar persones que busquen companys de conversa en funció d'interessos comuns o de proximitat.

7.7 El SoC ESP32

Quan es parla del circuit integrat ESP32 es fa referencia al SoC (System on Chip) de l'empresa Espressif que integra diferents components dins del mateix xip dels quals destaca el seu processador de 32 bits de doble nucli dotat de connectivitat WiFi i Bluetooth. Actualment aquest Soc conjuntament amb el ESP8266 són els més utilitzats en els projectes del IoT gracies al seu baix preu i les seves prestacions orientades a la connectivitat.

El processador d'aquest SoC és un Xtensa LX6 de Tensilica de 32 bits de doble nucli, a una velocitat de rellotge estàndard de 160 MHz que es pot arribar a programar fins els 240 MHz. A més, disposa de 520 kB de memòria RAM integrada accessible per als dos processadors, malgrat això podria ampliar-se fins els 8 MB.

Aquest SoC no inclou una memòria integrada de 448 kB utilitzada pel bootloader i les seves funcions internes. El SoC original no integra memòria flash però pot gestionar una flash externa de fins a 16 MB, dins de la família dels ESP32 existeixen diferents versions que varien en característiques alguns d'ells com el ESP32-WROOM-32 (que és el que es farà servir) inclouen memòria flash, en aquest cas 4 MB.

El SoC del ESP32 accepta una alimentació de 2,2 a 3,6 V i té un consum mig de 80 mA amb un màxim de 225 mA. Pot operar entre -40 Cº a 125 Cº de temperatura i accepta encriptació per hardware.

7.7.1 Connectivitat

Respecte a la connectivitat, disposa de connexió WiFi compatible amb els estàndards b/g/n i es pot configurar en mode estació (STA), punt d'accés (Soft AP) o mode combinat (AP + STA).

El mòdul ESP32 disposa dels següents blocs de radiofreqüència:

- Un receptor de 2,4 GHz que desmodula la senyal de RF per convertir-la al domini digital mitjançant dos convertidors A/D d'alta resolució. El integrat incorpora filtres RF, control automàtic del guany i filtres de banda base integrats.

- Un transmissor de 2,4 GHz encarregat de modular la senyal en banda base a senyal de RF modulada a 2,4 GHz i enviar-la a l'antena mitjançant un amplificador de potencia basat en CMOS. Aquest inclou uns calibratges integrats que permeten cancel·lar les adicions de soroll.
- Un rellotge que genera una senyal en quadratura de 2,4 GHz per la transmissió i la recepció de WiFi.

La connexió Bluetooth suportada es la versió 4.2 BR(*Basic Rate*) / EDR(*Enhanced Data Rate*) i disposa de BLE (*Bluetooth Low Energy*).

El mòdul ESP32 té 34 pins GPIO amb diferents funcions assignables, trobem pins digitals, digitals i analògics, i pins amb funció Touch. La majoria d'aquests pins es poden configurar en mode pull-up, pull-down o alta impedància.

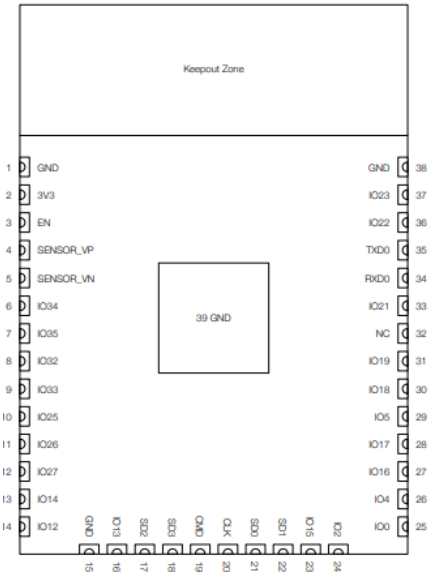
El ESP32 disposa de dos convertidors analògic-digital (ADC) de 12 bits capaç de funcionar quan el dispositiu entra en mode d'estalvi d'energia, gràcies al coprocessador de baix consum. També posseeix un convertidor digital-analògic (DAC) de 8 bits que permet convertir fins a dos senyals digitals en analògiques.

Compta amb una sèrie de sensors interns com un sensor de temperatura, un sensor d'efecte Hall, que detecta un camp magnètic prop del dispositiu, o els sensors tipus Touch en alguns pins, que poden detectar variacions a l'apropar o tocar el pin amb el dit o amb un objecte.

També es disposa d'un controlador SD / SDIO / MMC que suporta traspàs de dades des de dispositius externs de fins a 80MHz amb diferents busos: de 1bit, de 4bits i de 8bits.

Per a la comunicació de dades sèrie el dispositiu compta amb tres interfícies UART que proporcionen comunicació asíncrona de fins a 5 Mbps. Altres protocols de comunicació com I2C, I2S i SCI són suportats pel dispositiu.

Suporta dues interfícies I2C, que poden servir com a mestre o com a esclau a una velocitat de fins a 5Mhz. D'altra banda, suporta comunicacions I2S per dues interfícies amb controladors DMA dedicats. I a més posseeix un controlador d'infrarojos que suporta diversos protocols. El ESP32 també pot generar fins a setze canals PWM (Pulse Width Modulation) i posseeix una interfície per a la comunicació per bus CAN 2.0.



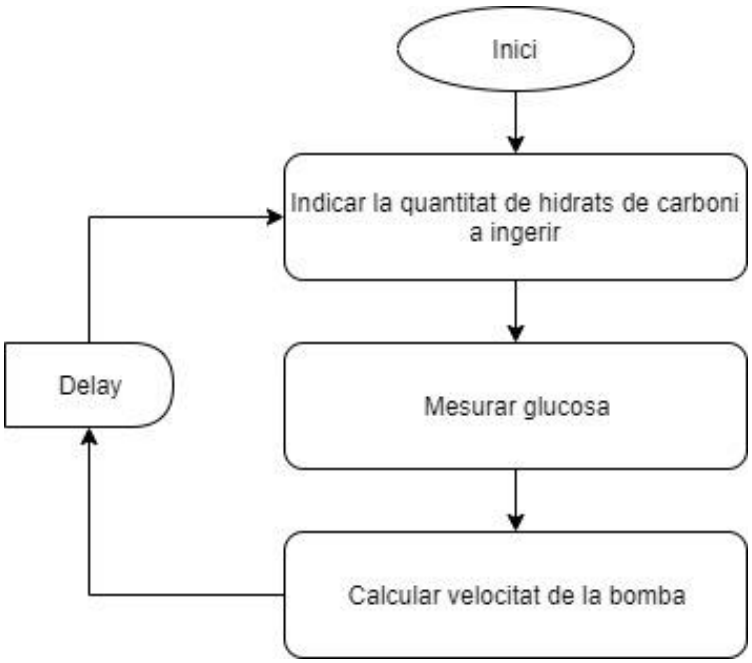
IL·LUSTRACIÓ 5. DISPOSICIÓ DELS PINS DEL ESP32-WROOM-32

.

8 El primer prototip

Per al primer prototip es fa servir una placa Arduino UNO amb un display i un LED que emularà la velocitat de la bomba. En aquest cas, els paràmetres venen ja definits i no existeix cap tipus d'interacció amb l'usuari.

Aquest prototip segueix una línia de funcionament molt bàsica:



ESQUEMA 1. ESQUEMA BÀSIC DE FUNCIONAMENT DEL PRIMER PROTOTIP

Principalment seguiria un esquema de funcionament d'un pàncrees artificial ideal, és a dir, l'usuari no tindria que preocupar-se de res, més enllà de mantenir l'equip amb bateria i realitzar manteniment periòdic del sensor i la bomba. No obstant, això s'allunya molt de la realitat, cap pacient pot tenir cada dia exactament sempre el mateix comportament i encara així els valors de glucosa no serien sempre iguals.

El principal problema de dissenyar un pàncrees artificial és la presa de decisions, ja que ni l'acció de la insulina ni la pujada del nivell de glucosa en sang quan s'ingereixen aliments és instantània, segueixen unes corbes d'acció, que en el cas de la insulina podríem considerar estables, però la glucosa en sang dependria molt del tipus, quantitat i combinacions d'aliments, on també hauríem d'afegir l'exercici i l'estat d'ànim del pacient com a variables.

Aquesta és la principal raó per la qual els actuals pàncrees artificials comercials requereixen d'una interacció amb el pacient i un exhaustiu seguiment i anàlisi previ per tal de que el dispositiu aporti una millora en la qualitat de vida del pacient.

Com l'objectiu principal d'aquest treball és dissenyar un sistema de control i afegir noves funcionalitats per aquests dispositius, es crea un primer prototip molt reduït en funcions. Això serà suficient per poder adaptar fàcilment el producte a altre pàncrees artificials comercials o de codi obert i només es gestionaran certs paràmetres bàsics.

8.1 Consideracions prèvies

Per el muntatge i disseny del primer prototip, s'han definit els patrons base que regiran el funcionament del pàncrees artificial. En el tractament de la diabetis tipus 1, a grans trets es poden utilitzar dos tipus de tractaments:

- L'administració d'insulina lenta que el cos anirà distribuint de manera basal durant unes hores determinades, junt a l'administració de insulina rapida distribuïda abans de la ingestió d'un àpat.
- La segona possibilitat es administrar contínuament uns nivells baixos d'insulina rapida per proveir al cos d'una quantitat d'insulina basal i pujar aquests nivells en els moments previs als àpats.

Un altre aspecte important a considerar es tracta del FSI (Factor Sensibilitat a la Insulina) que ens indica la quantitat de glucosa que es redueix per unitat d'insulina en un pacient. Aquest factor ha de ser establert per un metge per tal de poder calcular les dosis de insulina de manera convenient. En aquest prototip prendrem un valor de referencia estàndard que sol ser FSI = 50 mg/dl per unitat d'insulina.

S'estableix un patró estàndard de velocitat basal, en aquest cas, no es prendrà cap actuació sobre aquests valors ja que son pautes que ha de indicar un doctor i afavoreixen el correcte funcionament dels nivells d'insulina durant el dia. S'ha de tenir en compte que aquesta serà aproximadament la meitat de la insulina que rebrà el pacient al llarg del dia.

Es defineix el concepte de bolus, que correspon a les unitats d'insulina que s'han d'introduir durant la següent hora en concepte de la quantitat d'hidrats de carboni que es consumiran. Per això s'estableix un màxim de 200 grams d'hidrats de carboni al dia, que és el màxim que es recomana per una persona diabètica.

Podem calcular fàcilment les unitats de insulina necessàries durant la següent hora sabent que cada 10 grams d'hidrats de carboni, la glucosa pujarà entre 35-50 mg/dl com es una quantitat variable establirem que la pujada serà de 5 mg/dl per gram.

Els patrons aplicats estan detallats a la següent taula:

HORA	Basal	Bolus	Glucosa	Grams Hidrats de Carboni
0	0,9	0	0	0
1	0,9	0	0	0
2	0,9	0	0	0
3	0,9	0	0	0
4	0,9	0	0	0
5	0,9	0	0	0
6	0,9	0	0	0
7	0,9	0	0	0
8	0,65	2	100	20
9	0,65	0	0	0
10	0,65	3	150	30
11	0,65	0	0	0
12	0,65	0	0	0
13	0,65	7	350	70
14	0,65	0	0	0
15	0,65	0	0	0
16	0,65	0	0	0
17	0,65	3	150	30
18	0,9	0	0	0
19	0,9	0	0	0
20	0,9	0	0	0
21	0,9	5	250	50
22	0,9	0	0	0
23	0,9	0	0	0
TOTAL	19,1	20	1000	200

TAULA 5. PATRONS UTILITZATS AL PROTOTIP

Només s'han tingut en compte les ingestes d'hidrats de carboni en les variacions de la glucosa, això es una aproximació molt llunyana a la realitat, ja que cada pacient té uns

patrons molt variables depenent el dia, es per això que el primer prototip només treballa amb un patró definit que simula les ingestes diàries a hores predeterminades amb els corresponents bolus d'insulina.

Es prendran bàsicament com a referencia els patrons indicats a la taula que emulen el funcionament ideal d'un pàncrees artificial amb uns horaris preestablerts, un patró estàndard d'insulina basal i uns horaris i ingestes predeterminats.

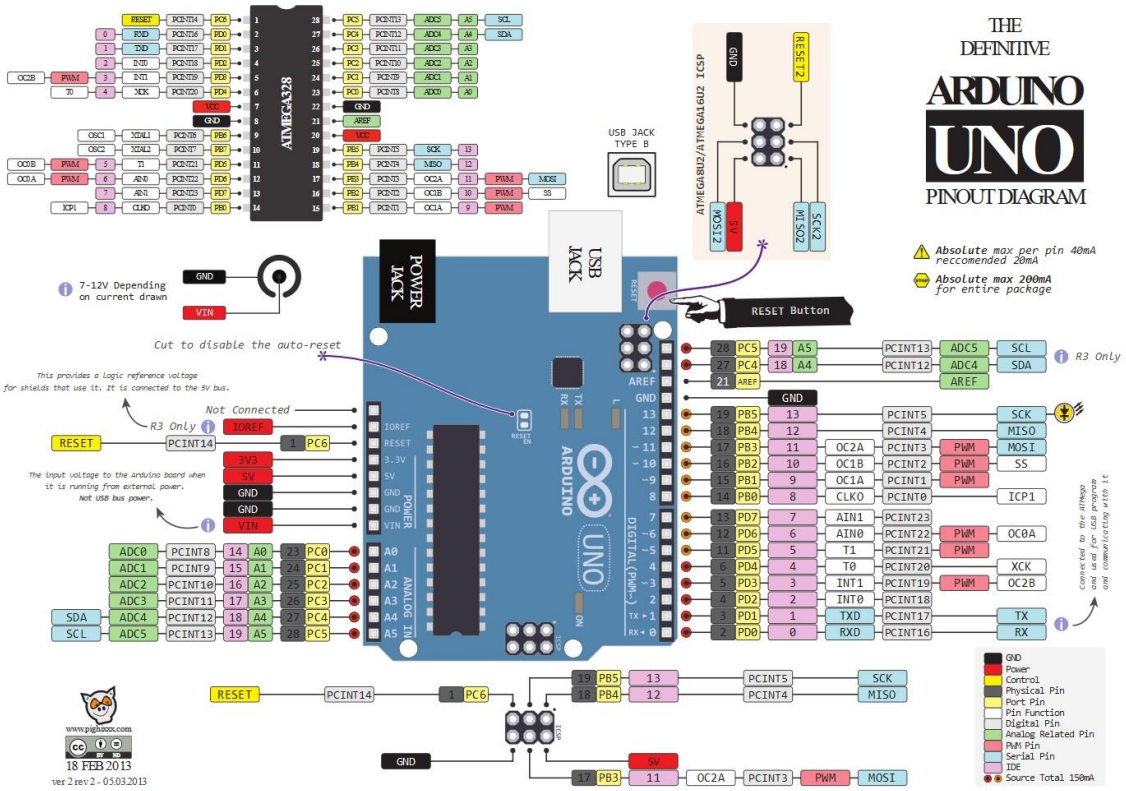
8.2 La placa Arduino UNO

En el muntatge del primer prototip es farà servir una placa Arduino UNO que serà l'encarregada de controlar, gestionar i alimentar els perifèrics. En aquest cas s'utilitzarà el IDE d'Arduino per controlar la programació i visualitzar la sortida del port sèrie.

Arduino UNO és una de les plaques de la família d'Arduino més utilitzades en l'actualitat del món *maker* i es pot trobar a infinitat de projectes. Les principals característiques d'aquesta placa són:

- Microcontrolador ATmega328P
- Tensió de funcionament 5V
- Voltatge d'entrada (recomanat) 7-12V
- Voltatge d'entrada (límit) 6-20V
- Digital pins I/O 14 (dels quals 6 proporcionen una sortida PWM)
- PWM digital pins I/O 6
- Pins d'entrada analògica juny
- Corrent DC per Pin I/O 20mA
- Corrent DC per Pin 3.3V 60mA
- Memòria flash 32KB ATmega328P dels que 0,5 KB són utilitzats pel gestor d'arrencada.
- SRAM 2KB ATmega328P
- EEPROM 1KB ATmega328P
- Velocitat de rellotge 16 MHz

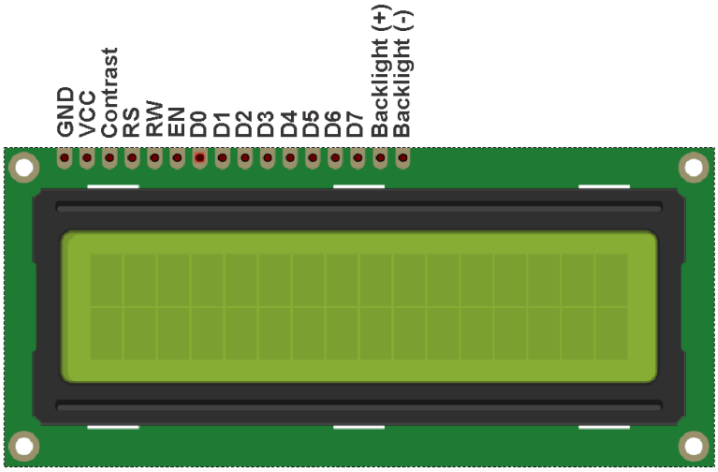
A continuació trobem una il·lustració amb el pinout detallat de la placa i el processador que facilita molt la programació i la connexió dels perifèrics i també el disseny dels circuits:



ESQUEMA 2. PINOUT DE LA PLACA ARDUINO UNO

8.3 La pantalla LCD de 16x2

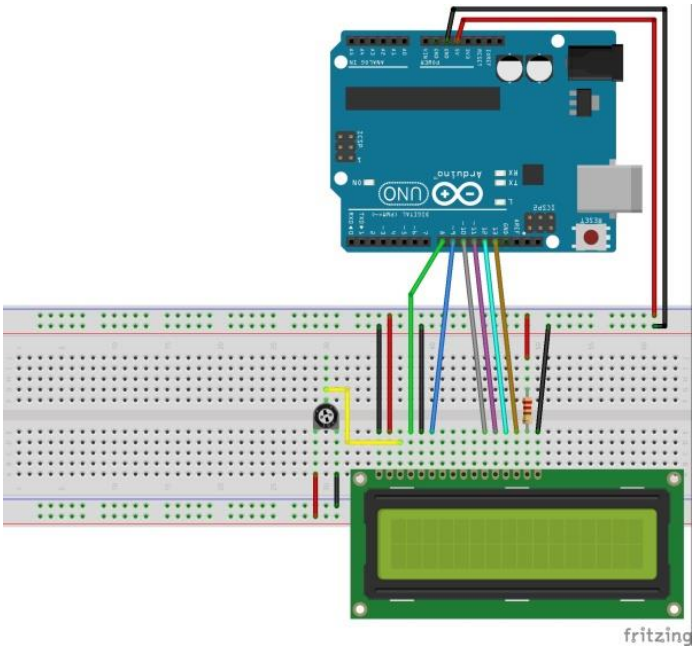
La forma més senzilla i còmoda per visualitzar les dades, és la connexió d'una pantalla LCD (*Liquid Cristal Display*). El model de pantalla 1602 fa servir una matriu de 16 caràcters x 2 línies on es posicionen els caràcters dins de la matriu de manera molt similar a com es faria amb la sortida serial gracies a la llibreria <LiquidCrystal.h>.



ESQUEMA 3. PINOUT MÒDUL DISPLAY 1602

La connexió de la pantalla es realitzarà mitjançant les següents consideracions:

- GND i VCC es connectaran directament a l'alimentació de 5V.
- Contrast es connectarà a l'alimentació i GND amb un potenciòmetre de 10 KΩ per poder modificar el contrast de la pantalla.
- RS és el selector de registre i ofereix informació al display per saber si s'envien comandes de control o si s'han de mostrar caràcters a la pantalla. Es connectarà al pin 8 d'Arduino.
- RW selecciona mode lectura o escriptura, es connectarà a GND per indicar que en tot moment es vol escriure a la pantalla.
- EN habilita la rebuda d'informació cap al display. Es connecta al pin 9 d'Arduino.
- D0 – D7 bus de dades. En aquest cas només es faran servir 4 bits d'informació D4 – D7 que es connectaran als pins 10 -13.
- Els pins de Backlight proporcionen la retro il·luminació i aniran connectats a VCC i GND. En aquest cas es farà servir una resistència de 220 Ω per ajustar la intensitat del LED d'il·luminació.



ESQUEMA 4. CONNEXIONS PANTALLA LCD AMB ARDUINO UNO

Un cop muntat el circuit es crea l'objecte lcd de la classe LiquidCristal i es passen els pins de connexió. Un cop creat l'objecte el funcionament és molt similar al de la sortida serial d'Arduino. Per inicialitzar el display es farà servir el mètode lcd.begin(16, 2) per tal de indicar que el display consta de 16 columnes i 2 files.

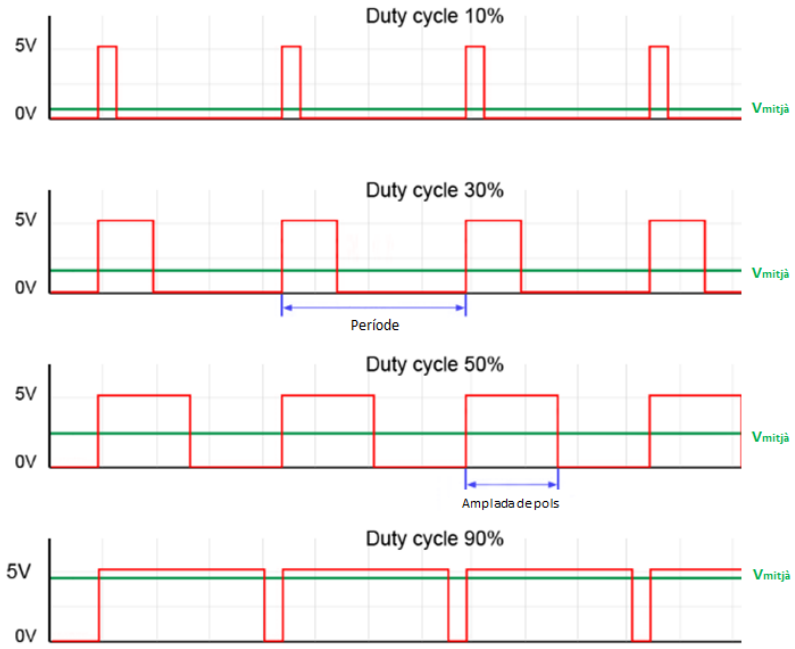
El posicionament del text es realitza a traves del mètode lcd.setCursor(x, y), el mètode lcd.print() imprimeix caràcters a pantalla i lcd.clear() esborra la pantalla.

8.4 El LED controlat per PWM

Les plaques Arduino, no tenen la capacitat de proporcionar una sortida analògica real, ni tan sols una sortida analògica discreta (generada amb salts de tensió), l'única opció possible es proporcionar una sortida digital que variï entre –Vcc i +Vcc, és per això que es fa servir la modulació per amplada de pols.

La modulació PWM (*Pulse Width Modulation*) en comptes de generar una senyal continua de sortida, genera una sèrie de polsos de durada variable però de intensitat constant. La mitjana de la tensió a la sortida, espaiada en el temps, serà equivalent al

valor analògic que es vol emular. La proporció de temps que la senyal es troba a nivell alt s'anomena cicle de treball o *DutyCycle* expressat en percentatge.



IL·LUSTRACIÓ 6. EXEMPLES DE SORTIDES PWM

Aquests paràmetres es poden calcular fàcilment mitjançant les següents expressions:

$$V_{mig} = (V_{cc+} - V_{cc-}) \cdot \frac{DutyCycle}{100}$$

$$DutyCycle = 100 \cdot \frac{V_{mig}}{(V_{cc+} - V_{cc-})}$$

Amb Arduino UNO es disposen de 6 sortides PWM amb una resolució de 8 bits, és a dir, que hi han 2^8 = 256 valors assolibles i es podrà escollir un valor per el DutyCycle entre 0 i 255. Per tal de generar polsos de llum amb diferents intensitats, es connecta el LED a la sortida 3 d'Arduino.

Primer es defineix el pin on es realitzarà la connexió, el pin 3 fent servir la declaració #define BOMBA 3. En el setup es configura el pin com sortida amb el paràmetre pinMode (BOMBA, OUTPUT).

Un cop configurat el LED es quantifica la intensitat que mostrarà fent servir la comanda map per establir uns valors d'intensitat entre 0 i 255, establirem que la velocitat de la bomba pot arribar com a màxim a 20 unitats de insulina per hora i multiplicarem la velocitat de la bomba per 10 per tal de que els decimals siguin representatius, finalment la comanda correspondria amb intensitat = map((velBomba * 10), 0, 100, 0, 255).

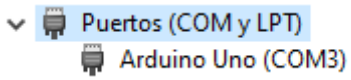
Un cop calculada la intensitat, aquesta serà traslladada al LED que hem establert com sortida i definit prèviament com connectat al pin 3. L'ordre analogWrite(BOMBA, intensitat) estableix aquest comportament.

El muntatge d'aquesta part es realitza connectant una resistència en sèrie amb el LED, s'escull una resistència de 220 Ω, ja que tenint en compte una caiguda de tensió al LED de 3.4 V, ens queden 1.6 V a la resistència, amb aquests valors es pot determinar una intensitat de corrent màxima de uns 73 mA ja que es tracta d'un circuït sèrie.

$$I = \frac{V_R}{R} = \frac{1.6}{220} = 0.007272\ A$$

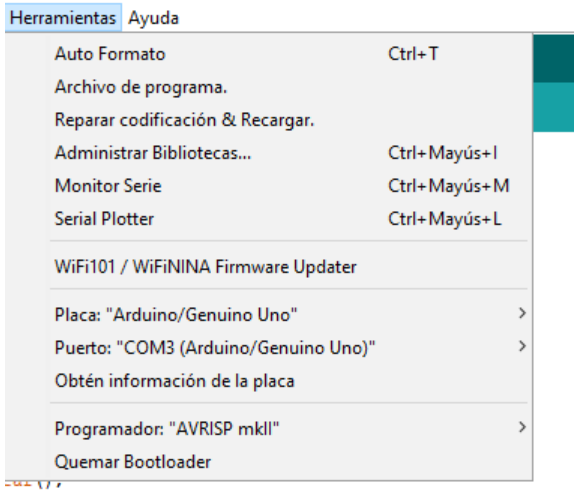
9 Programació i muntatge del primer prototip

Per la programació de la placa s'utilitzarà l'IDE oficial d'Arduino, en primer lloc hem de cercar al administrador de dispositius el port COM de connexió de la placa Arduino amb l'ordinador.



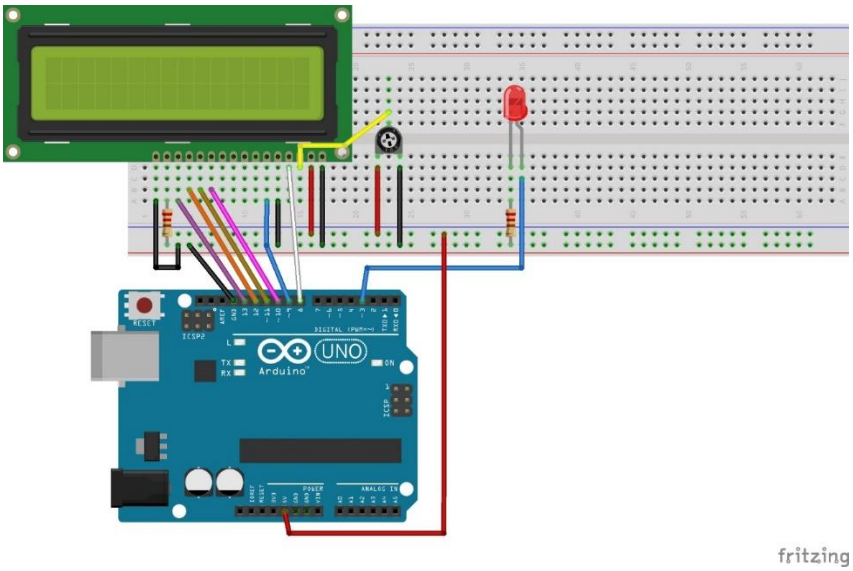
IL·LUSTRACIÓ 7. PORT COM ASSIGNAT

Un cop determinat el port, en aquest cas el COM3 passem a configurar l'IDE per la programació de la placa, on només ens queda configurar el model de la placa en aquest cas s'escull Arduino / Genuino UNO.



IL·LUSTRACIÓ 8. CONFIGURACIÓ PER TREBALLAR AMB LA PLACA ARDUINO UNO

Determinades les opcions només resta seleccionar l'Sketch (codi del programa) i programar-lo a la placa. El muntatge físic del circuit es pot observar en el següent esquema:



ESQUEMA 5. MUNTATGE DEL PRIMER PROTOTIP

10 El segon prototip

Per aquest prototip s'ha optat per fer servir una placa de desenvolupament, basat en un SoC de ESP32 i es farà us de un bot de Telegram per tal de indicar les ingestes, obtenir alarmes i consultar les dades. A la banda d'usuari s'afegeix un display OLED connectat per I2C per la visualització de les dades i un LED que simuli la velocitat de la bomba en cada moment.

En aquest segon prototip s'implementa també un LED amb intensitat variable mitjançant PWM amb un procediment igual que el emprat en el primer prototip i com es pot observar el codi es el mateix i el muntatge molt similar, el que dona una visió de la escalabilitat dels dispositius Arduino i similars.

A més, s'implementa una rutina de seguretat per controlar l'accés al bot i es crearan dos rols d'accés al bot. El primer rol correspon al pacient que serà l'usuari que tindrà accés a totes las opcions del bot, el segon rol comptaria amb menys opcions i s'anomenarà familiar.

En un principi es comptarà amb la possibilitat de un pacient i dos familiars, encara que aquesta opció es pot modificar en un futur i adaptar a les necessitats de cada pacient. Les opcions que disposa cada familiar també es podran modificar segons les necessitats del client. A més s'afegiran alarmes per a tots els usuaris per mitja de la missatgeria de Telegram.

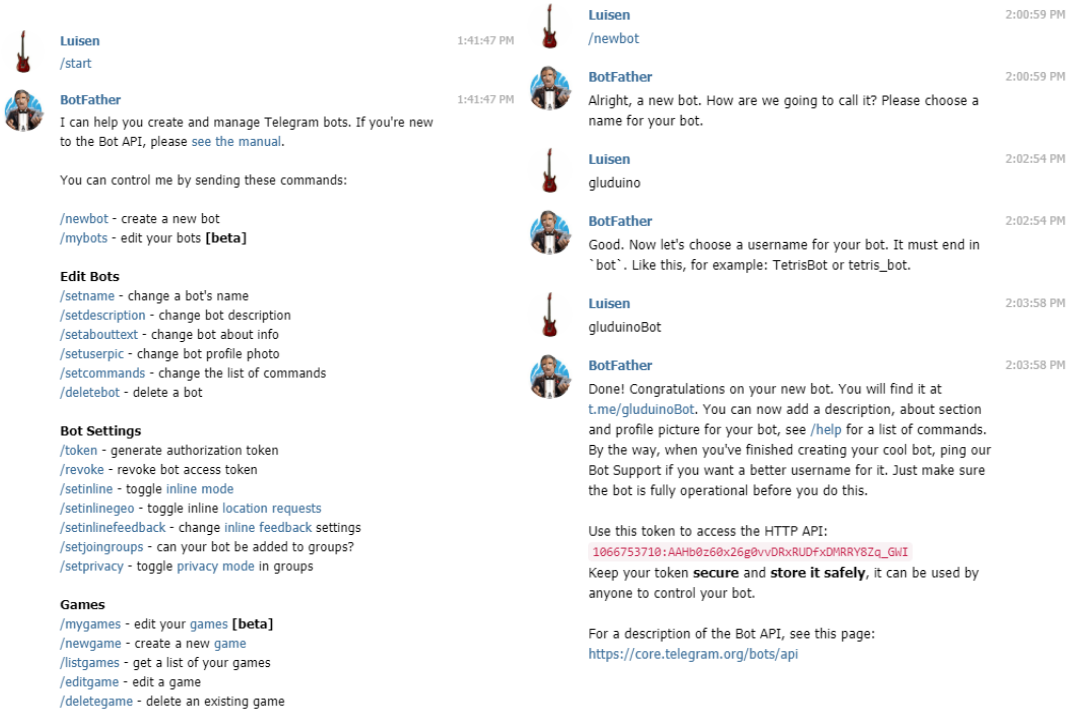
Les rutines de control de glucosa i de la velocitat de la bomba, restaran similars a les utilitzades en el primer prototip amb la gran diferencia de a que les ingestes prenen un caire interactiu amb l'usuari. El comportament de la insulina basal seguirà el mateix procediment que en l'anterior prototip, s'establiran unes pautes predefinides que les quals no es tindran en compte en la emulació de la glucosa.

10.1 Creació del Bot de Telegram

Qualsevol usuari de Telegram, té la possibilitat de crear un bot. La creació de un bot de Telegram és un procés molt senzill a la par de curiós. En primer lloc, l'usuari ha de

contactar amb el @BotFather, un altre bot que com el seu nom indica és el pare de tots els bots, i invocar la comanda /start.

Un cop iniciat el bot, aquest deixa triar una sèrie de paràmetres, en aquest cas es farà servir /newbot per crear un nou bot i seguidament el bot demanà el nom del bot a crear i després el nom d'usuari del bot, que obligatòriament ha d'acabar amb la cadena de text 'bot'. Si els noms triats estan disponibles es crea el bot i ens facilita el token i unes indicacions bàsiques. A continuació trobem unes captures amb el procés de creació del bot emprat en aquest prototip.

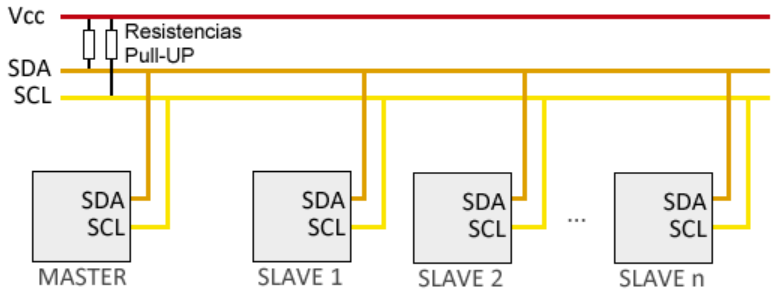


IL·LUSTRACIÓ 9. CAPTURA DE LA CREACIÓ DEL BOT DE TELEGRAM

Un cop realitzat el procediment, el @BotFather ens retorna el següent token: `1066753710:AAHb0z60x26g0vvDRxRUDfxDMRRY8Zq_GWI` que és el que necessitarem per accedir al nostre bot mitjançant la API HTTP de Telegram.

10.2 El bus I2C

L'estàndard I2C (Inter-Integrated Circuit) va ser desenvolupat per Philips al 1982 principalment per tal de facilitar la comunicació entre els seus propis dispositius, però poc a poc va ser adoptat per els diferents fabricants fins que va esdevenir un estàndard a la industria. La connexió entre els dispositius es realitza en paral·lel i la comunicació entre ells es sèrie.



ESQUEMA 6. CONNEXIÓ BUS I2C

Aquest bus només necessita de 2 punts de connexió per el seu funcionament, un per l'enviament de dades SDA (Serial DAta) i un altre per l'enviament de la informació de rellotge SCL (Serial CLock). I2C consta d'un direccionalment de 7 bits per tant es poden arribar a connectar 27-1 dispositius, és a dir 127, ja que el direccionalment 0 es fa servir per fer una crida general a tots els dispositius connectats al bus.

El bus I2C té una arquitectura del tipus Master – Slave. El dispositiu establert com Master és l'encarregat d'establir la connexió amb els Slaves, aquests no poden iniciar la comunicació, han de ser requerits per el Master, els Slaves tampoc es poden comunicar entre ells directament. Encara que no s'implementa normalment degut a la complexitat del sistema, existeix la possibilitat de la existència de més d'un Master a un bus I2C però només pot exercir aquest rol un a la vegada.

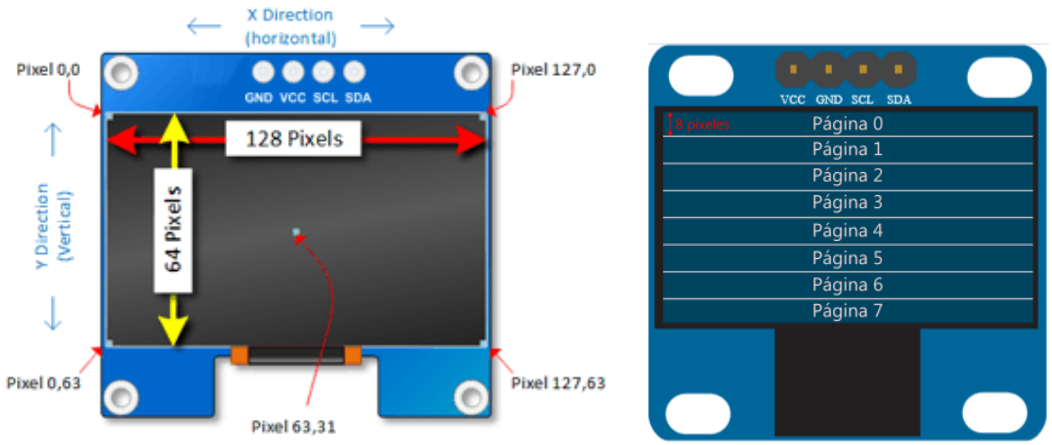
Per una correcta implementació es requereixen unes resistències de Pull Up amb la línia de Vcc, un factor que s'ha de gestionar amb molta cura en el cas de requerir connexions d'alta velocitat amb un màxim de 1000 Kbits/s, o millorar la fiabilitat en llargues distancies de comunicació. En el cas de dispositius Arduino o similars la llibreria

<Wire.h> gestiona aquest protocol i s'encarrega d'administrar les resistències internes, en altres dispositius és necessari revisar la documentació del fabricant.

10.3 El display OLED

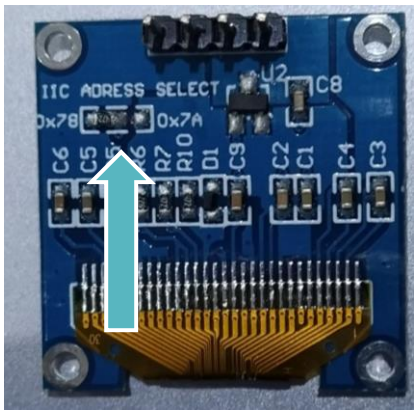
En aquest prototip es fa servir una pantalla OLED (Organic Light-Emitting Diode) de 0,96" amb una resolució de 128x64 píxels, model SSD1306 controlada a través del bus I2C de la placa. Aquesta pantalla pot operar en un rang de voltatges entre 3 – 5 V i té un consum molt baix, 15 mA.

La pantalla té una resolució de 128 columnes de 64 píxels, que per facilitar l'escriptura es distribueix en 8 pagines de 8 píxels d'alçada cadascuna:



ESQUEMA 7. PARÀMETRES BASICS DEL DISPLAY

Per tal de poder treballar amb el port I2C del display, es important saber la adreça que té aquest per poder establir una correcta sincronització. A més, aquest display a la part posterior disposa de una resistència que pot ser soldada a un altre pont per tal de canviar aquesta adreça en cas de coincidència amb altre dispositiu I2C.



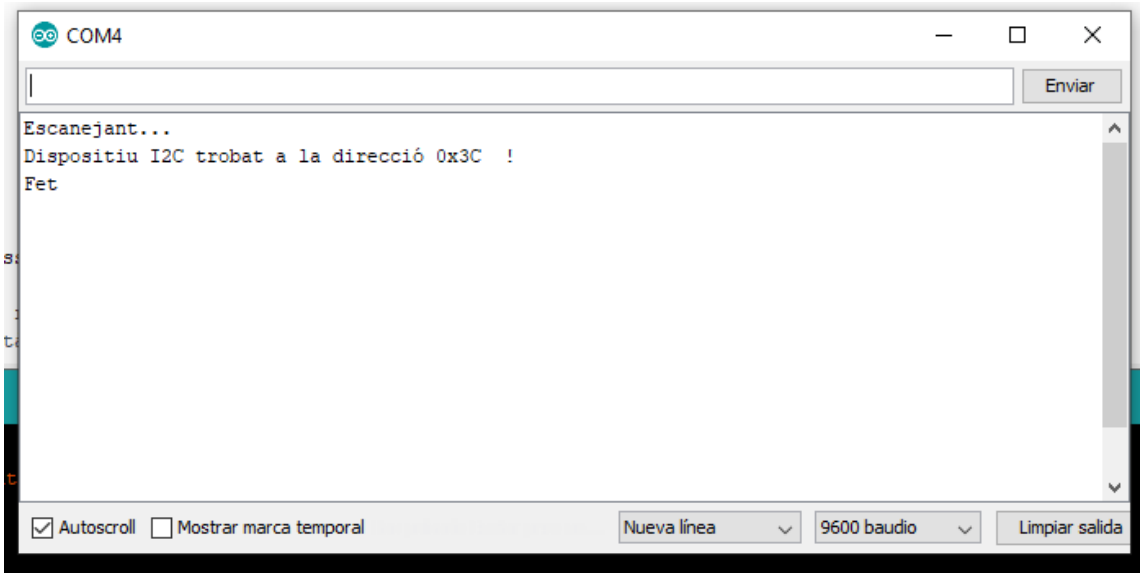
IL·LUSTRACIÓ 10. SELECCIÓ DE LA ADREÇA I2C

Una solució per trobar l'adreça I2C del display es pot torbar executant un petit Sketch que ens retornarà l'adreça fent servir el port sèrie:

```
#include <Wire.h>

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  while (!Serial);             // Espera al monitor sèrie
  Serial.println("\nEscàner I2C");
}

void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Escanejant...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // L'escàner i2c utilitza el valor retornat per la instrucció
Wire.endTransmisstion
    // per reconèixer la direcció on està connectat cada dispositiu.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)
    {
      Serial.print("Dispositiu I2C trobat a la direcció 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Error desconegut a la direcció 0x");
      if (address<16)
```

```
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("Cap dispositiu I2C trobat\n");
  else
    Serial.println("Fet\n");

  delay(5000);           // Espera 5 segons per el següent escaneig
}
```

Un cop executat l'Sketch s'obté el s'obté la direcció 0x3C com es pot observar a la captura del monitor sèrie:



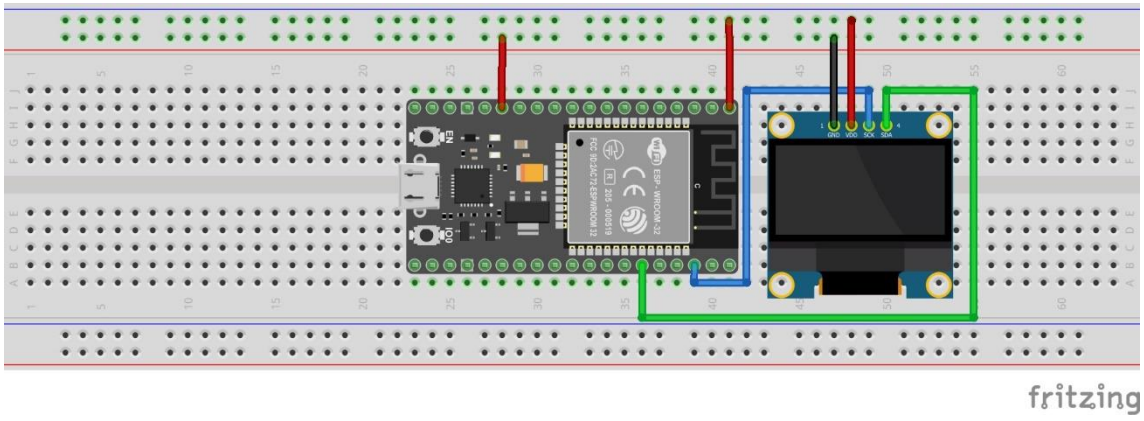
IL·LUSTRACIÓ 11. RESULTAT ESCANEIG I2C

Amb aquesta dada ja es possible treballar amb el display OLED, i per això son necessàries 3 llibreries, la primera <Wire.h> per poder controlar el bus I2C, <Adafruit_SSD1306.h> és un controlador per pantalles OLED model SSD1306 de Adafruit i finalment s'inclou la llibreria <Adafruit_GFX.h> per facilitar opcions gràfiques més elaborades.

Per poder enviar dades fàcilment al display primerament es crea un objecte display del tipus Adafruit_SSD1306 i se li passen els arguments d'amplada i llargada de la pantalla,

les dades del controlador I2C i el pin de reset Adafruit_SSD1306 amb la ordre display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET).

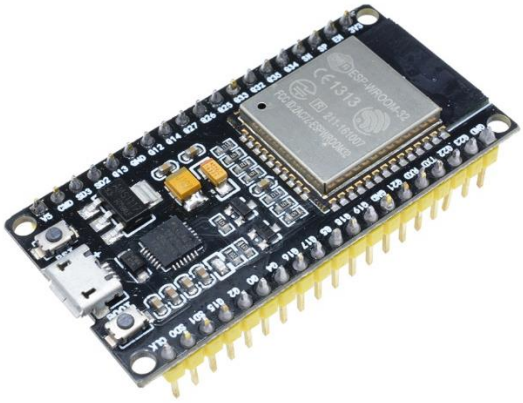
Un cop realitzada la definició només resta inicialitzar el display passant l'adreça I2C de la pantalla display.begin(SSD1306_SWITCHCAPVCC, 0x3C). A partir d'aquest punt es poden passar ordres al display com .clearDisplay(), print(), println() o ordres per modificar les característiques de les fonts setTextSize(), setTextColor() o setCursor() per posicionar a un punt de la pantalla, entre d'altres. Un cop formatat el contingut que es vol enviar a la pantalla l'enviarem a traves del mètode display.display().



ESQUEMA 8. ESQUEMA DE CONNEXIÓ DEL DISPLAY OLED

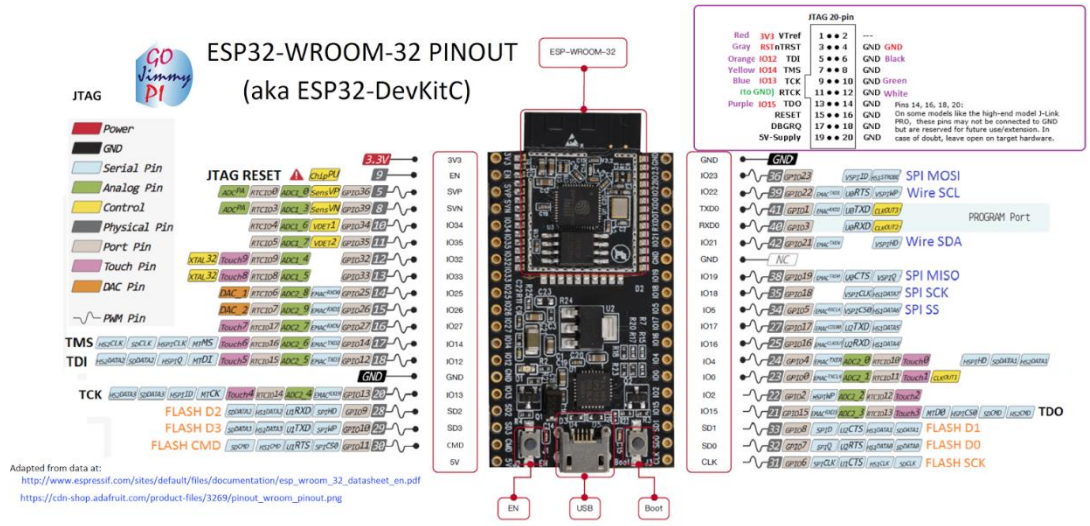
10.4 La placa de desenvolupament ESP32 DevKit

La targeta de desenvolupament empleada per aquest prototip, correspon amb el model GOOUUU-ESP32 que disposa del SoC ESP-WROOM-32 i inclou interfície USB que a més, serveix per l'alimentació de la placa i uns pins adaptats per a unes connexions més senzilles.



IL·LUSTRACIÓ 12. TARGETA DE DESENVOLUPAMENT ESP32-WROOM-32

Aquesta targeta inclou el xip CP2102 que s'encarrega de fer la conversió de serial o TTL a USB, amb una velocitat de connexió que va des de els 300 bps a 1 Mbps. També disposa de dos interruptors, un de reset i l'altre de boot que resulta molt pràctic per pujar els programes a la placa.



ESQUEMA 9. PINOUT DE LA PLACA ESP32-WROOM-32

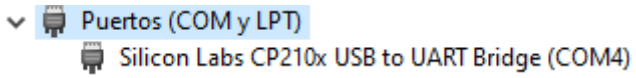
Encara que en aquest projecte s'ha utilitzat aquest model de targeta, una senzilla recerca per internet permet observar que existeixen una gran varietat de plaques de desenvolupament de ESP32 algunes inclús amb el display ja integrat o un porta bateries per una pila 18650 a uns preus molt assequibles.

10.5 Configuració del IDE per a la placa ESP32

El SoC ESP32 pot ser programat en diversos llenguatges LUA, micropython, Arduino, Espressif IDF o Javascript entre d'altres. En aquest cas es farà sevir el IDE d'Arduino i serà necessari realitzar algunes instal·lacions i configuracions prèvies ja que d'entrada l'IDE no reconeixerà la placa.

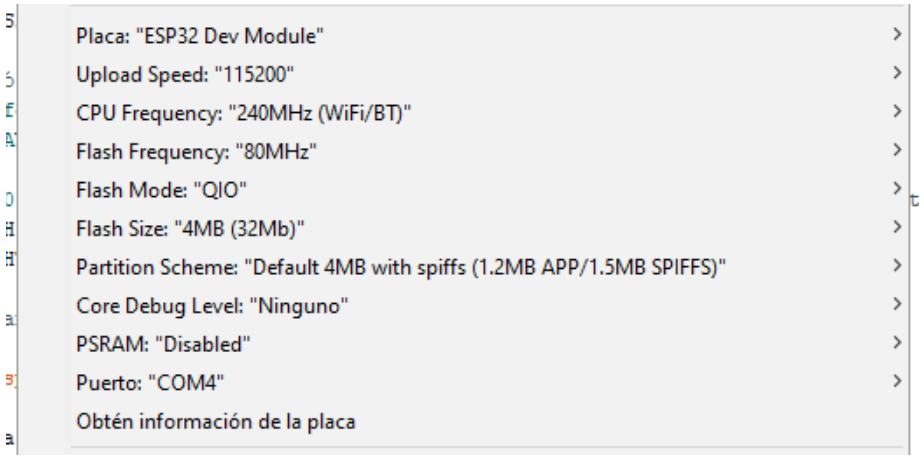
Per la programació del ESP32 primer s'ha d'accedir al repositori de Espressif a Github i mitjançant el Git GUI es clona el repositori en el directori de instal·lació del IDE d'Arduino. Un cop realitzat aquest pas s'ha d'instal·lar el compilador que ve inclòs en el repositori.

Un cop descarregat el repositori s'ha de procedir a la configuració del IDE per la placa, en primer lloc s'ha de trobar el port on s'ha connectat la placa, es pot revisar fàcilment al administrador de dispositius del sistema.



IL·LUSTRACIÓ 13. PORT COM ASSIGNAT

Un cop definit el port, s'escullen les següents opcions per poder programar la targeta de desenvolupament del ESP32:



IL·LUSTRACIÓ 14. CONFIGURACIÓ PER TREBALLAR AMB LA PLACA ESP32 DEVKIT

Un cop realitzada la configuració del IDE ja és possible treballar amb la placa de la mateixa manera que l'Arduino UNO amb la diferencia que en el moment de la programació, és a dir, carregar l'Sketch a la placa, un acabada la compilació s'ha de pitjar en botó de BOOT per entrar en mode programació.

10.6 La comunicació amb la API de Telegram

En el segon prototip és on es troben les funcions més interesants que requereixen de la comunicació de la placa amb el bot de Telegram. La API de Telegram és la encarregada de dur a terme aquesta tasca, i per facilitar aquesta feina existeixen diverses llibreries que s'encarreguen de fer de nexe entre la API i la placa.

S'ha de tenir clar que en realitat un bot, a priori, no està allotjat als servidors de Telegram, aquesta és una plataforma de missatgeria i només emmagatzemarà els missatges i arxius enviats fins al moment que aquests s'esborrin i donarà certes funcionalitats de *Front End*.

És a dir, es necessari que la placa estigui connectada a internet per tal de poder fer servir el bot, si no, el missatge quedarà emmagatzemat esperant la connexió de la placa per enviar-se. Un cop rebut el missatge, la placa el processarà i realitzarà les accions corresponents a la comanda rebuda.

Telegram funciona amb una xarxa de servidors distribuïts, que es comuniquen mitjançant un protocol propi de codi obert amb una encriptació AES amb claus de 256 bits anomenat *MTProto* (*Mobile Transport Protocol*). A més, fa servir el protocol d'intercanvi de claus privades/publiques *Diffie-Hellman*, i els missatges es signen digitalment amb les claus privades.

La API és l'encarregada de interactuar amb el bot, els bots han estat dissenyats per ocultar tota la part del protocol de xifrat *MTProto*. Per tant permetrà la comunicació amb els servidors de Telegram intercanviant senzills missatges HTTPS.

Per tant, amb unes reduïdes dades bàsiques es podrà interactuar amb Telegram mitjançant diverses llibreries:

- El token del bot, que es el identificador únic que el @BotFather ens ha assignat.
- El chatID, cada cop que s'inicia una conversa Telegram genera un chatID únic que serà necessari per accedir al xat.

10.7 La librería UniversalTelegramBot.h

Per poder establir comunicació amb els servidors de Telegram a la banda de la placa, on s'allotja el bot, són necessàries tres llibreries, una de forma directa i les altres de manera indirecta a través de la primera llibreria, aquestes son UniversalTelegramBot.h, ArduinoJson.h i WifiClientSecure.h.

En primer lloc, s'ha de tenir en compte que la llibreria UniversalTelegramBot.h necessita una versió concreta de la llibreria ArduinoJson.h en aquest cas s'ha d'instal·lar-se la versió 5.13.4 al gestor de llibreries del IDE.

Aquesta dependència es necessària ja que com s'ha comentat abans el protocol empleat per comunicar-se amb la API de Telegram en HTTPS però la informació que contindran aquests paquets serà en format Json i és aquesta llibreria la encarregada de estructurar els missatges Json degudament.

La llibreria WifiClientSecure.h és l'encarregada d'aplicar el protocol SSL (Secure Sockets Layer) i proveirà els certificats als paquets HTTPS que s'enviaran a la API a través de la llibreria UniversalTelegramBot.h.

```
#define BOTtoken
"1066753710:AAHb0z60x26g0vvDRxRUDfxDMRRY8Zq_GWI"
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
```

Per la creació del bot de Telegram el primer pas seria crear un objecte UniversalTelegramBot al que se li passarà el token i el client SSL. Un cop creat el bot ja està preparat per rebré comandes externes i el següent pas a realitzar serà el disseny de les rutines per a la gestió de missatges.

10.8 La gestió de missatges de Telegram

El primer punt importat en la gestió de la recepció de missatges de Telegram és que es necessita una rutina que treballi de forma concurrent amb els altres processos, per això no es fa servir un delay, si no, que es recorre a la funció millis() que compta el numero de mil·lisegons que fa que s'ha engegat la placa.

```
if (millis() > Bot_lasttime + Bot_mtbs)  {
  int numNewMessages = bot.getUpdates(bot.last_message_received
+ 1);

  while(numNewMessages) {
    Serial.println("got response");
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received +
1);
  }
  Bot_lasttime = millis();
}
```

Es a dir, fins que el valor de la funció millis() no arribi fins el temps que resulta de la suma de l'últim instant executat, més el temps en mil·lisegons definit a la variable Bot_mtbs s'executarà la rutina i en el cas de que arribi un missatge es gestionaran per la rutina handleNewMessages(numNewMessages) a la qual li passarem el nombre de missatges rebuts.

10.9 El mètode handleNewMessages()

Aquest mètode és un dels més importants del bot, podria considerar-se el cervell d'aquest. És l'encarregat de gestionar tots els missatges. El mètode handleNewMessages() rep com argument el nombre de nous missatges rebuts i els gestiona de forma seqüencial.

Cada missatge rebut, té unes propietats que són les que s'utilitzaran per tal de gestionar-los. Les principals propietats que és faran servir son:

- *.chat_id* retorna el chatID on pertany el missatge.

- *.from_name* retorna el nom de l'emissor del missatge establert a la plataforma de Telegram.
- *.text* retorna el text del missatge.

Així doncs es guarden aquestes propietats en noves variables que s'utilitzaran principalment en la pressa de decisions del bot. Dins d'aquest mètode també s'han implementat les rutines de seguretat que s'establiran en l'accés i que veurem detallades més endavant.

Posteriorment s'analitza la cadena de text rebuda i es gestiona de manera literal, o tenint en compte algun argument necessari en la resposta de bot. Això es pot implementar fent servir el mètode substring() per separar l'argument i així poder avaluar-lo de manera independent a la ordre rebuda.

```
if (text == "/velocitat") { … } // Avaluació literal
```

```
if (text.startsWith("/menjar")) { //Avaluació amb argument
        String arg = text.substring(7);
… }
```

Un cop que ja s'ha generat la resposta del bot, només resta un últim pas que correspon amb l'enviament de la resposta a la API de Telegram mitjançant el mètode .sendMessage i rep com argument en chatID i el text a enviar.

```
bot.sendMessage(chat_id, texto);
```

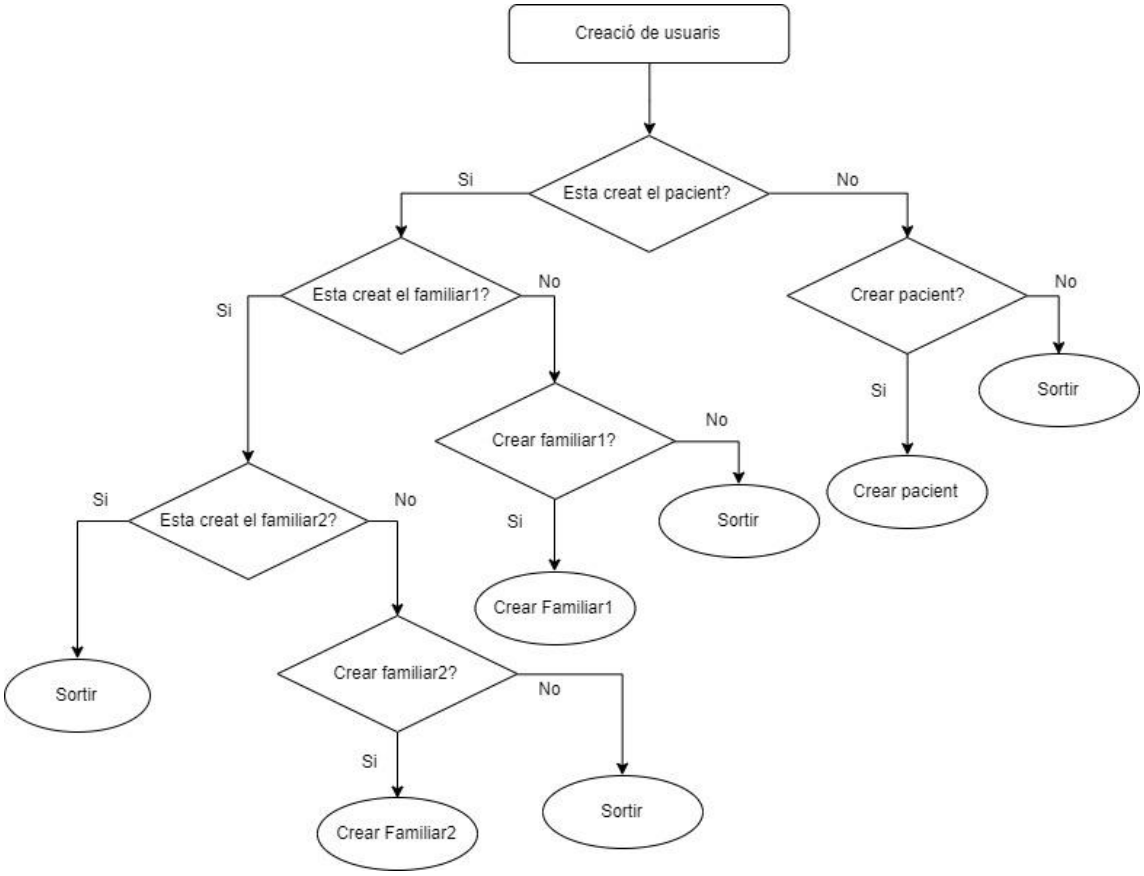
10.10 Gestió de seguretat d'accés

Com es comentava anteriorment es necessària la creació de diferents rols d'accés, dependents del tipus d'usuari que vol accedir al sistema. També és un punt important controlar qui pot i qui no pot accedir al sistema per tal d'evitar l'entrada al bot a persones que no han estat prèviament enregistrades.

En primera instancia el bot permet el registre de tres usuaris diferents, un pacient i dos familiars, el primer en registrar-se s'exigirà que sempre sigui el pacient, en cas contrari

no deixarà continuar, i els següents perfils que es podran configurar al sistema seran els dos familiars.

A continuació es pot observar un esquema de les possibles opcions disponibles en el procés de la creació dels usuaris:



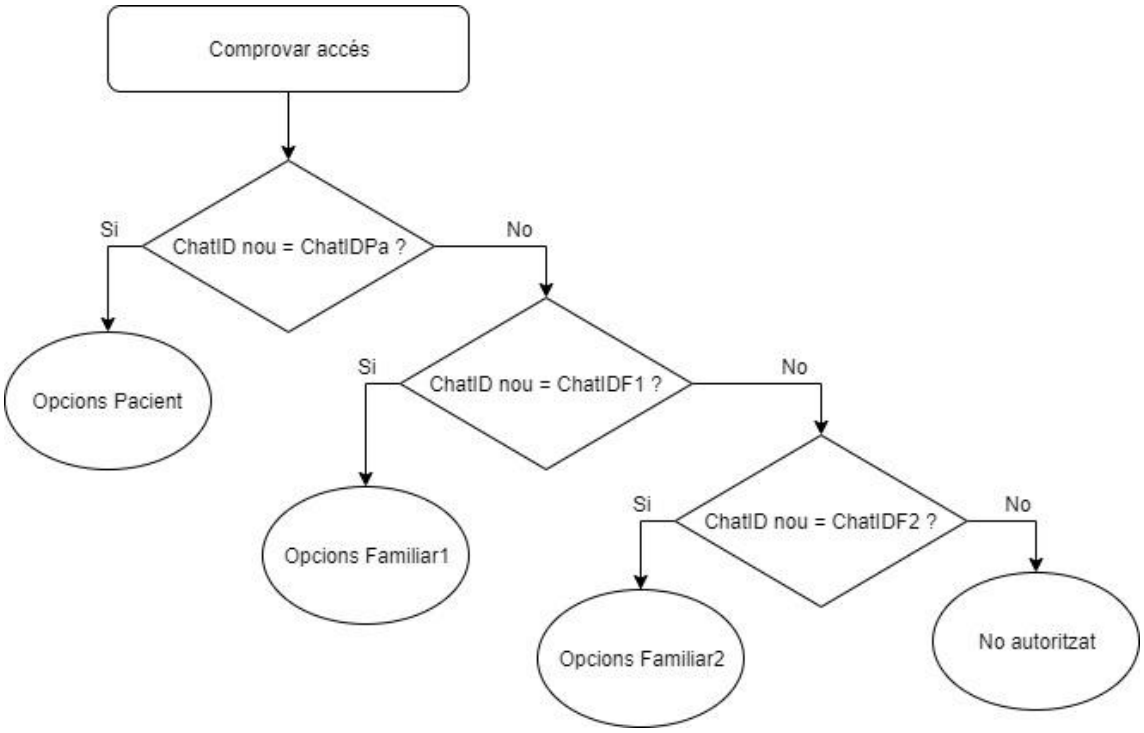
ESQUEMA 10. DIAGRAMA DE CREACIÓ DE USUARIS

Aquest criteri seguit en la creació d'usuaris podria ser diferent en futures revisions, ja que d'entrada no es un sistema de registre persistent, en un futur s'ha de preveure la creació de una base de dades externa on guardar els usuaris, de no ser així les configuracions es perdran cada cop que el bot faci un reset.

Una altra possible solució al problema de la persistència seria enregistrar prèviament al codi als usuaris del bot, el problema vindria en el moment que es volgués fer alguna actualització d'usuaris, ja que hauríem de tornar a pujar el codi a la placa.

En l'enregistrament dels usuaris, en realitat el que es fa és una revisió dels chatID, de la conversa actual amb el bot, dels usuaris que han accedit i es creen tres variables on s'emmagatzemaran els usuaris registrats. Un cop guardades es comparen amb les dels missatges que han entrat i per seguretat només deixa accedir si el chatID correspon amb el requerit.

El següent diagrama mostra el flux de dades en el control d'accés:



ESQUEMA 11. DIAGRAMA DE CONTROL D'ACCÉS

10.11 El mode gràfic

A mode d'exemple, s'implementa un mode gràfic que ens mostrarà un diagrama de barres molt simple amb els valors de la velocitat de la bomba durant les últimes 24 hores, que ens pot donar una pauta orientativa de l'ús de la bomba.

Per aquest objectiu es recorre a la funció .drawRect() proporcionada per la llibreria de Adafruit que necessita els paràmetres display.drawRect (X_0, Y_0, amplada, alçada, color) seguidament amb un for es recorreran els valors del array durant les 24 posicions per poder representar totes les barres.

10.12 Les alarmes

Encara que un cop implementat tot el codi i disposant dels chatIDs necessaris les alarmes són una de les funcions més senzilles, també esdevenen una de les més importants, ja que es la funcionalitat estrella que haurien de disposar tots els dispositius d'aquest tipus.

Les alarmes estableixen un control extra molt important que esdevé decisiu en el cas dels pacients més dependents i ofereixen mes tranquil·litat a les famílies que estarien informades en tot moment o inclús donaria sentit a la creació d'un valor afegit que seria el seguiment per part d'un grup de professionals que tinguessin cura del pacient i actuessin en cas d'emergència.

El funcionament no pot ser més senzill:

ESQUEMA 12. DIAGRAMA DE ENVIAMENT D'ALARMES

La programació és molt senzilla, amb un if s'avalua una variable, en aquest cas la glucosa, i si el valor contingut per aquesta està fora del criteri establert s'enviarà un missatge via Telegram al pacient i/o als familiars, segons el criteri establert per l'usuari.

Val a dir, que les programacions en un futur han de ser personalitzades ja que cada pacient te les seves pròpies particularitats, depenent de quins tipus de necessitats tinguin el pacient, les alertes potser no interessa que les rebin tots els usuaris del sistema.

El mètode emprat per l'enviament d'alertes via Telegram és .sendMessage(), que rep dos paràmetres, el chatID del xat a enviar el missatge i òbviament el missatge a enviar, és a dir bot.sendMessage(chatID, missatge).

11 Futures línies de treball

En relació a aquest projecte, podria dir-se que queda molta feina a fer i moltes millores a realitzar. Però les característiques que considero més interesants de cara a un futur pròxim seria treballar el tema de la localització, treballar el tema de la persistència, l'addició d'una base de dades externa i implementar un sistema de RTC (Real Time Clock) per tenir un control horari real.

El primer punt, la geolocalització, és una afegit molt important en persones dependents, ja que permetrien geolocalitzar al malalt en cas d'emergència. Es poden trobar fàcilment dues opcions valides per implementar aquesta funcionalitat:

- El primer mètode passaria per afegir un mòdul de GPS extern a la placa, el qual ens proveiria de la informació de latitud i longitud que fàcilment es podria codificar directament en una URL amb la ubicació a Google Maps. Amb aquesta solució s'augmentaria el preu i la mida del prototip.
- La segona opció consistiria en utilitzar la geolocalització WiFi, que consisteix en utilitzar la potencia que ens arriba de cada encaminador proper per calcular la distancia aproximada amb aquest. Aquestes dates juntament amb les MACs dels encaminadors s'enviarien a la *Google Maps GeoLocation API* que retornaria la ubicació a Google Maps. Amb aquesta solució es necessari un compte de *Google* i utilitzar la seva plataforma de desenvolupament *Firebase*. A més, haurem de tenir en compte que segons l'ús que es faci podria requerir una subscripció de pagament.

Seria interesant també treballar el tema de la persistència, ja que com es comentava anteriorment, les solucions passarien per integrar els usuaris registrats en el codi o crear una base de dades externa. Si es vol obtenir versatilitat la línia de treball haurà de seguir la segona opció.

A més, la opció del disseny d'una base de dades anònima externa possibilitaria en un futur una gran sistema de Open Data que podria ser un complement molt interessant en futures investigacions.

Per últim la implementació de un horari real ens aproparia el prototip a una funcionalitat real, tot i que, el mode de treball actual amb la emulació del temps sempre resulta molt pràctic a mode de depuració i es podrien crear dues línies de treball independents.

12 Conclusions i resultats

Finalment s'ha realitzat el disseny del dos prototips amb un correcte funcionament, malgrat això personalment m'hauria agradat disposar de molt més temps, doncs trobo que tres mesos no és temps suficient temps per desenvolupar un sistema tan delicat com seria un pàncrees artificial. Es per això que només s'ha pogut dissenyar el mòdul de control i encara així jo ho visualitzaria com una base del sistema de control final.

Hagués estat interesant treballar una mica més els algoritmes de control de la glucosa però el desconeixement del camp de la endocrinologia i els complexes patrons que segueixen els nivells de glucosa a la sang conjuntament amb en comportament variable de la insulina en l'organisme al cap de varies hores fan que s'incrementi molt la complexitat dels algoritmes.

En un principi es va treballar en la emulació de un sensor de glucosa amb un fotoresistor, i un LED com el generador de glucosa del sistema. Poc a poc, aquesta part va deixar de tenir massa sentit en el prototip ja que és una emulació que no ens aporta cap millora al projecte. De tota manera va fer el seu servei en el punt de conèixer una mica millor el comportament dels perifèrics a la placa.

Per altra banda, crec que el tema de les comunicacions ha estat molt satisfactori. El Bot de Telegram finalment ha resultat bastant més avançat que el que en un principi havia pogut imaginar i he adquirit bons coneixements del seu funcionament visualitzant totes les capes d'abstracció d'aquest.

El funcionament, així com el muntatge dels displays dels dos prototips ha estat satisfactori i encara que no s'ha aprofundit s'han implementat algunes funcions gràfiques bàsiques que obren la porta a una interfície d'usuari més amigable en un futur.

Així com l'ús de diferents tipus de busos en les plaques basades en Arduino ha facilitat una simplificació important en el muntatge del segon prototip el que incrementa la viabilitat del projecte ja que fora interessant reduir al màxim la mida dels nostres prototips.

Per ultima instancia, s'han implementat de forma satisfactòria les alarmes, per mi un tema basic i un mínim imprescindible per que el projecte adquireixi un sentit real.

També la implementació de la seguretat en l'accés i la gestió d'usuaris ha estat un èxit tot i no ser presents en els plantejaments inicials.

S'ha de ser sincer, i la correcta planificació i l'organització del projecte ha estat un punt feble d'aquest treball i la idea inicial i una línia de treball errònia han enrederit moltes parts i a afavorit moltes modificacions a l'ultima part del treball.

Com una conclusió final, ha estat un treball que ha canviat molt si ens basem en el plantejament inicial que pretenia enfocar-se molt més en el pàncrees artificial. Podem dir que ha evolucionat més cap al desenvolupament del mòdul de control i s'ha enfocat en les comunicacions com a part principal.

13 Bibliografia

Adafruit. (2012, 29 julio). Monochrome OLED Breakouts. Recuperat de https://learn.adafruit.com/monochrome-oled-breakouts/arduino-library-and-examples

Adafruit, A. (s.f.). adafruit/Adafruit-GFX-Library. Recuperat de https://github.com/adafruit/Adafruit-GFX-Library

Arduino - Home. (s.f.). Recuperat de https://www.arduino.cc/

Arduino - Products. (s.f.). Recuperat de https://www.arduino.cc/en/Main/Products

Arduino Uno Pinout, una sencilla introducción a su esquema. (2019, 9 mayo). Recuperat de http://www.electrogeekshop.com/arduino-uno-pinout-una-sencilla-introduccion-a-su-esquema-5-52/

Arduino Uno, partes, componentes, para qué sirve y donde comprar. (2019, 25 octubre). Recuperat de https://descubrearduino.com/arduino-uno/

Bots: An introduction for developers. (s.f.). Recuperat de https://core.telegram.org/bots

CDC. (s.f.). Información sobre la diabetes. Recuperat de https://www.cdc.gov/diabetes/spanish/basics/diabetes.html

Colaboradores de Wikipedia. (2019, 8 enero). Plataforma abierta de hardware y software. Recuperat de https://es.wikipedia.org/wiki/Arduino

Colaboradores de Wikipedia. (2019, 8 diciembre). Interfaz de usuario para bots en Telegram Messenger. Recuperat de https://es.wikipedia.org/wiki/Telegram_Bot_API

Colaboradores de Wikipedia. (2020, 3 enero). protocolo exclusivo para clientes Telegram y afines optimizada en el envío de mensajes. Recuperat de https://es.wikipedia.org/wiki/MTProto

Dani No, D. N. (2016, 14 noviembre). PRÁCTICA 5: Funcionamiento de una pantalla OLED con el procesador ESP8266. Recuperat de https://www.esploradores.com/practica-5-funcionamiento-de-una-pantalla-oled-con-el-procesador-esp8266/

Ecarletti, E. (2018, 7 diciembre). Descripción y funcionamiento del Bus I2C | Robots Didácticos. Recuperat de http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/

Espressif Systems. (s.f.). ESP32 Overview | Espressif Systems. Recuperat de https://www.espressif.com/en/products/hardware/esp32/overview

Fritzing. (s.f.). Fritzing. Recuperat de https://fritzing.org/home/

Fundación para la Diabetes. (s.f.). Bomba de insulina. Recuperat de https://www.fundaciondiabetes.org/infantil/185/bomba-de-insulina-ninos

Fundación para la Diabetes. (2014, 29 enero). ¿Cuánto sube la glucemia una ración de hidratos de carbono? Recuperat de https://www.fundaciondiabetes.org/general/articulo/19/cuanto-sube-la-glucemia-una-racion-de-hidratos-de-carbono

Ganttproject. (s.f.). Ganttproject. Recuperat de https://www.ganttproject.biz/

Jaime Recarte, J. R. (2019, 16 marzo). Así es el páncreas artificial para controlar la glucosa. Recuperado 4 enero, 2020, de https://www.redaccionmedica.com/secciones/tecnologia/asi-es-el-pancreas-artificial-que-controlara-automaticamente-la-diabetes-2751

Luis Llamas, L. L. (2017, 30 octubre). El bus I2C en Arduino. Recuperat de https://www.luisllamas.es/arduino-i2c/

Luis Llamas, L. L. (2018, 1 abril). ESP32, el "hermano mayor" del ESP8266 con WiFi y Bluetooth. Recuperat de https://www.luisllamas.es/esp32/

Luis Llamas, L. L. (2019, 7 diciembre). Salidas analógicas PWM en Arduino. Recuperat de https://www.luisllamas.es/salidas-analogicas-pwm-en-arduino/

Mario Hernandez Luria, M. H. L. (s.f.). Cómo funciona un medidor de glucosa | Asvidia. Recuperat de https://www.asvidia.org/como-funciona-un-medidor-de-glucosa/

Medtronic. (s.f.). ¿Qué es una terapia con bomba de insulina? Recuperat de https://www.medtronic-diabetes.com/es/terapia-con-bomba-de-insulina

Prometec. (2019, 8 julio). Indice de tutoriales Arduino | Tienda y Tutoriales Arduino. Recuperat de https://www.prometec.net/indice-tutoriales/

Ramiro Antuña de Alaiz, R. A. A. (2012, 19 marzo). Ajuste diario de las insulinas utilizando el factor de sensibilidad. Recuperat de https://clinidiabet.com/es/infodiabetes/educacion/tratamiento/insulina/12.htm

Telegram – a new era of messaging. (s.f.). Recuperat de https://www.telegram.org/

The Internet of Things with ESP32. (s.f.). Recuperat de http://esp32.net/

Tobo, T. (2019, 2 octubre). ESP32 Pinout Reference: Which GPIO pins should you use? Recuperat de https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

Tr4nsduc7or, T. (2018, 11 marzo). Como conectar una pantalla OLED SSD1306 con Arduino sin librerías | robologs. Recuperat de https://robologs.net/2018/03/11/como-conectar-una-pantalla-oled-ssd1306-con-arduino-sin-libreria/

Víctor Guillem, V. G. (2015, 15 julio). Creando un BOT con el API de Telegram I. Recuperat de https://www.genbeta.com/desarrollo/creando-un-bot-con-el-api-de-telegram-i

Witnessmenow, W. (s.f.). witnessmenow/Universal-Arduino-Telegram-Bot. Recuperat de https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot

Yúbal FM, Y. F. M. (2018, 3 agosto). Qué es Arduino, cómo funciona y qué puedes hacer con uno. Recuperat de https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

14 Annexos

14.1 Codi del primer prototip

```
#include <LiquidCrystal.h> //Llibreria per fer servir el display
#define BOMBA 3 //Pin on es connecta el LED que emula la bomba

LiquidCrystal lcd(8,9,10,11,12,13); //Es defineixen pins de connexió del lcd
int gluIni = 120;
int glucosa;
float velHora[] =
{0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.9,0
.9,0.9,0.9,0.9,0.9};
float bolus[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; //Matriu on es
guardaràn els bolus
float hidrats[] = {0,0,0,0,0,0,0,0,100,0,150,0,0,350,0,0,0,150,0,0,0,250,0,0};
//Quantitat de glucosa que aportaran els hidrats
int FSI = 50;
byte hora;
float velBomba;

void setup() {

          Serial.begin(115200); // configurem la velocitat del port serie
          pinMode (BOMBA, OUTPUT); //indiquem que el port es de sortida
          glucosa = gluIni; // S'inicialitzen les variables
          hora = 0;
          velBomba = 0;
          lcd.begin(16, 2); // Numero de caracters i files
          lcd.print("Pancreas Ver 1.0"); // Enviar el missatge
          delay(2000);
}

void loop() {

          menjar(hora); //Es comprova si hi ha augment de glucosa
          mesuraGlucosa(hora); // Es mesura glucosa teorica
          velBomba = calculaVel(hora); // Es calcula velocitat de la bomba
          activaBomba();
          impDisplay();

          delay(2000);

          if (hora == 23) hora = 0;
          else hora++;
}

void activaBomba(){
          int intensitat = map((velBomba * 10), 0, 100, 0, 255);
          analogWrite(BOMBA, intensitat);
}

void impDisplay(){
          //S'imprimeix la informació al display LCD
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Vel bomba ");
```

```
          lcd.print(velBomba);
          lcd.setCursor(0,8);
          lcd.print("Glucosa ");
          lcd.print(glucosa);
}

float calculaVel(int h) {

          bolus[h] = hidrats[h] / FSI;
          float velocitat = velHora[h] + bolus[h];
          return velocitat;

}

void mesuraGlucosa(int h) {

          if (h > 1){
          if (bolus[h-1] > 0){
          glucosa = glucosa - (int(bolus[h-1]) * FSI);
                    }
          }
}

void menjar(int h) {

          glucosa = glucosa + hidrats[h];
}
```

14.2 Codi del segon prototip

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <analogWrite.h> // Llibreria per fer servir funció analogWrite amb ESP32

// Dades de connexió WIFI
char ssid[] = "vodafone1B02";     // SSID
char password[] = "AVCMR2YCYLVSJ6"; // Contrasenya

#define BOTtoken "1066753710:AAHb0z60x26g0vvDRxRUDfxDMRRY8Zq_GWI" // Bot Token obtingut
al Botfather
#define SCREEN_WIDTH 128 // OLED display amplada, en pixels
#define SCREEN_HEIGHT 32 // OLED display llargada, en pixels
#define BOMBA 19 //Pin on es connecta el LED que emula la bomba

// Declaration del display SSD1306 connectat per I2C
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

String texto, alerta;
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

int Bot_mtbs = 1000; // Temps d'escaneig entre misatges
long Bot_lasttime;   //Variable on es guarda el darrer instant de l'ultim escaneig
String chat_idp = "";
String chat_idf1 = "";
String chat_idf2 = "";
bool grafic = false;
bool bolusMenja = false;

float gluIni = 120;
float glucosa;
float velHoraBasal[] =
{0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.65,0.9,0
.9,0.9,0.9,0.9,0.9};
float velHoraDia[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float gluHoraDia[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float menjarHora[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float FSI = 50;
int glucoXhidrat = 5;
byte hora;
float velBomba;


void setup() {

      Serial.begin(115200);
      display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // //Inicialitza pantalla a l'adreça
0x3C del port I2C
            display.clearDisplay();
      Serial.print("Connectant Wifi: ");
      Serial.println(ssid);

      // Connecta el WIFI en mode client
```

```
      WiFi.mode(WIFI_STA);
      WiFi.begin(ssid, password);
      // Dibuixa punts fins que s'estableix la connexió
      while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
      }
      // Informació de la correcta connexió al WIFI
      Serial.println("");
      Serial.println("WiFi connectat!!");
      Serial.print("Adreça IP: ");
      Serial.println(WiFi.localIP());

      glucosa = gluIni;
      hora = 0;
      velBomba = 0;
      pinMode (BOMBA, OUTPUT); // Indiquem que el port es de sortida

            display.clearDisplay();//Netejem la pantalla
            display.setTextSize(1);
            display.setTextColor(1,0);
            display.setCursor(0,0);
            display.println("Gluduino Ver 1.0");
            display.display(); //Refresca la pantalla amb les noves dades
            delay(4000);
    Serial.println("Pancreas Ver 1.0");
}

void loop() {


  resetMenjar(); // Posem a 0 el hidrats de la següent hora
  calculaVel(hora); // Establim la velocitat de la bomba
  activaBomba(); // Emula la velocitat de la bomba amb la intensitat del LED
  calculaGlu(); // Fem el calcul teoric de glucosa
  mostraDisplay(); // Actualitza pantalla
  avisa(); // Es generen avisos

  Serial.print("Vel bomba ");
  Serial.println(velBomba);
  Serial.print("Glucosa ");
  Serial.println(glucosa);

  if (millis() > Bot_lasttime + Bot_mtbs)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    // Gestió dels nous missatges
    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }

    Bot_lasttime = millis();
  }


  delay (5000);
  guardaDades(); //Guardarem els valors de la glucosa i dels bolus d'insulina de cada
hora
  calculaGluFinal();
  if (hora == 23) hora = 0;
  else hora++;
```

```
}

void mostraDisplay(){
  if (grafic == false){
  display.fillScreen(0);          //Netejem la pantalla
  display.setRotation(0);
  display.setTextSize(1);
  display.setTextColor(1,0);
  display.setCursor(0,0);
  display.print("Velocitat bomba: ");
  display.println(velBomba);
  display.print("Glucosa: ");
  display.println(glucosa);
  display.print("\nHora: ");
  display.println(hora);
  display.display();              //Passem les dades al display
  } else{
         display.clearDisplay();
    display.setRotation(2);
      for (int j=0 ; j<24 ; j++){

           int v = int(velHoraDia[j]);
           display.drawRect( 10+(4*j), 2 , 4 , (v*3), 1 );
        }
      display.display();
  }

}

void handleNewMessages(int numNewMessages) {
  Serial.println("Gestió de misatges \nMisatges rebuts:");
  Serial.println(String(numNewMessages));

  // Gestió dels missatges rebuts seqüencialment
  for (int i=0; i<numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    String text = bot.messages[i].text;

    String from_name = bot.messages[i].from_name;
    if (from_name == "") from_name = "Guest";
    // Gestió de les opcions
    if (text == "/start") {

      if (chat_idp == ""){
                  texto =  "Benvingut a Gluduino, " + from_name + ".\n";
          texto += "Tria una opció:\n\n";
          texto += "/Pacient \n";
          texto += "/Familiar \n";
      }
      else if (chat_idp != "" && (chat_idf1 == "" || chat_idf2 == "")){
          texto =  "Benvingut a Gluduino, " + from_name + ".\n";
          texto += "Tria una opció:\n\n";
          texto += "/Familiar \n";
      }
      else if (chat_idp != "" && chat_idf1 != "" && chat_idf2 != ""){
          texto = "Ho sento, no estàs autoritzat";
      }
    }
    else if (text == "/Pacient" || text == "/pacient"){
         if (chat_idp == ""){
         chat_idp = chat_id;
```

```
        texto =  "Benvingut a Gluduino, Pacient\n";
        texto += "Tria una opció:\n\n";
        texto += "/menjar (Quantitat d'hidrats de carboni)\n";
        texto += "/glucosa\n";
      texto += "/velocitat\n";
        texto += "/gluh\n";
      texto += "/velh\n";
      texto += "/grafic\n";
        } else texto = "Opció no disponible";
  }
  else if (text == "/Familiar" || text == "/familiar"){
        if (chat_idp == ""){
                  texto = "Primer has de configurar el pacient";
        } else {
                  if (chat_idf1 == ""){
                            chat_idf1 = chat_id;
                            texto =  "Benvingut a Gluduino, Familiar 1\n";
                            texto += "Tria una opció:\n\n";
                            texto += "/menjar (Quantitat d'hidrats de carboni)\n";
                            texto += "/glucosa\n";
                            texto += "/velocitat\n";
                            texto += "/gluh\n";
                          texto += "/velh\n";
                  }
                  else if (chat_idf1 != "" && chat_idf2 == ""){
                            chat_idf2 = chat_id;
                            texto =  "Benvingut a Gluduino, Familiar 2\n";
                            texto += "Tria una opció:\n\n";
                            texto += "/menjar (Quantitat d'hidrats de carboni)\n";
                            texto += "/glucosa\n";
                            texto += "/velocitat\n";
                            texto += "/gluh\n";
                          texto += "/velh\n";
                  }
                  else if (chat_idp != "" && chat_idf1 != "" && chat_idf2 != ""){
                            texto = "Ho sento, no estàs autoritzat";
                  }
        }
  }
  else if (text == "/Opcions" || text == "/opcions"){
        if (chat_id == chat_idp){
                  texto += "Tria una opció:\n\n";
                  texto += "/menjar (Quantitat d'hidrats de carboni)\n";
                  texto += "/glucosa\n";
                texto += "/velocitat\n";
                texto += "/gluh\n";
                texto += "/velh\n";
                texto += "/grafic\n";
                texto += "/opcions\n";
        }
        else if (chat_id == chat_idf1 || chat_id == chat_idf2){
                  texto += "Tria una opció:\n\n";
                  texto += "/glucosa\n";
                texto += "/velocitat\n";
                texto += "/gluh\n";
                texto += "/velh\n";
                texto += "/opcions\n";
        }
  }
  else if (text == "/velocitat") {
        if (chat_id == chat_idp || chat_id == chat_idf1 || chat_id == chat_idf2){
                  texto = "Velocitat de la bomba: "+ String(velBomba);
```

```
          } else texto = "Ho sento, no estàs autoritzat";
    }
    else if (text == "/glucosa") {
          if (chat_id == chat_idp || chat_id == chat_idf1 || chat_id == chat_idf2){
                    texto = "La glucosa actual és: "+ String(glucosa);
          } else texto = "Ho sento, no estàs autoritzat";
    }
    else if (text.startsWith("/menjar")) {
          if (chat_id == chat_idp){
                    String arg = text.substring(7);
                    float argf = arg.toInt();
                    float bol = (argf * glucoXhidrat) / FSI;
                    texto = "S'aplicaràn " + String(bol) + " unitat(s) d'insulina";
                    aplicaBolus(argf);
          } else texto = "Ho sento, no estàs autoritzat";
    }
    else if (text.startsWith("/gluh")) {
          if (chat_id == chat_idp || chat_id == chat_idf1 || chat_id == chat_idf2){
                    String arg = text.substring(5);
                    int hor = arg.toInt();
                    if (hor >= 0 && hor <= 23){
                              texto = "La glucosa a les " + String(hor) + " ha sigut: " +
String(gluHoraDia[hor]) + " mg/dl";
                    }else texto = "Parametre no compatible";
          } else texto = "Ho sento, no estàs autoritzat";
    }
    else if (text.startsWith("/velh")) {
          if (chat_id == chat_idp || chat_id == chat_idf1 || chat_id == chat_idf2){
                    String arg = text.substring(5);
                    int hor = arg.toInt();
                    if (hor >= 0 && hor <= 23){
                              texto = "La velocitat de la bomba a les " + String(hor) + "
ha sigut: " + String(velHoraDia[hor]) + " u/h";
                    }else texto = "Parametre no compatible";
          } else texto = "Ho sento, no estàs autoritzat";
    }
    else if (text == "/grafic"){
          if (chat_id == chat_idp){
        if (grafic == false) texto = "Entrant en mode grafic";
        else texto = "Sortint mode grafic";
        grafic = !grafic;
          }else texto = "Ho sento, no estàs autoritzat";
    }
    else {
          texto = "Opció no disponible";
    }
    bot.sendMessage(chat_id, texto); // S'envia missatge a Telegram
   }
}

void activaBomba(){
          int intensitat = map((velBomba * 10), 0, 100, 0, 255);
          analogWrite(BOMBA, intensitat);
}

void resetMenjar(){
          if (hora !=23)
                    menjarHora[hora + 1] = 0;
          else if (hora == 23)
                    menjarHora[0] = 0;
}
```

```
void calculaVel(int h) {

          velBomba = velHoraBasal[h] + ((menjarHora[h] * glucoXhidrat)/FSI);

}

void calculaGlu() {

          glucosa = glucosa + (menjarHora[hora] * glucoXhidrat);
}

void avisa(){

          if (glucosa > 200) {
          alerta = "Atenció, la glucosa a les " + String(hora) + " està alta: "+
String(glucosa) + " mg/dl";
          if (chat_idp != "")bot.sendMessage(chat_idp, alerta);
          if (chat_idf1 != "") bot.sendMessage(chat_idf1, alerta);
          if (chat_idf2 != "") bot.sendMessage(chat_idf2, alerta);
          }
}

void calculaGluFinal(){

          glucosa = glucosa - ((velBomba - velHoraBasal[hora])*FSI);
}

void aplicaBolus(float u){

          if (hora !=23)
                    menjarHora[hora + 1] = menjarHora[hora + 1] + u;
          else if (hora == 23)
                    menjarHora[0] = menjarHora[0] + u;
}

void guardaDades() {

          gluHoraDia[hora] = glucosa;
          velHoraDia[hora] = velBomba - velHoraBasal[hora];
}
```