UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

EQUIP DIDÀCTIC DE TESTEIG DEL BUS CAN EN ENTORNS INDUSTRIALS AMB ALT CONTINGUT D'EMISSIONS ELECTROMAGNÈTIQUES

Memòria i Annexos

Autor: Roger Curià Castellnou
Director: Alfonso Conesa Roca
Departament: EEL

Convocatòria: Gener 2020

Resum

El present treball te com objectiu la construcció d'un equip didàctic que implementi una xarxa d'àrea de controladors basada en el protocol BUS CAN (sigles de l'anglès *Controller Area Network*) sobre la que es pugui analitzar el seu rendiment en entorns industrials amb altes emissions electromagnètiques.

El sistema s'ha implementat mitjançant plaques de desenvolupament de hardware, en aquest cas d'Arduino®, controladors CAN comercials i perifèrics d'entrada/sortida que permeten gestionar les funcions bàsiques de l'equip. La xarxa CAN està constituïda per dos nodes de transmissió de dades y el bus de comunicacions estandarditzat per al protocol CAN. Cadascun dels nodes està format per un microprocessador programable, un teclat i/o *display* i un mòdul controlador/transceptor CAN que el connecta al bus de comunicacions. El *software* desenvolupat en aquest treball facilita la gestió del sistema, les funcions de transmissió de dades i la visualització dels resultats de les comunicacions a través de la xarxa CAN.

Les proves de laboratori realitzades en aquest treball han posat de manifest la robustesa del protocol CAN i la seva idoneïtat per utilitzar-lo en entorns industrials sorollosos amb reduïda càrrega d'intercanvi de dades entre els diferents controladors i sensors del sistema.

Amb l'elaboració d'aquest treball, es pretén formar una base sobre la qual recolzar-se a l'hora d'utilitzar o d'implementar projectes amb la tecnològica CAN, desglossant-ne una competència que presenti solucions i tècniques per a la implementació, abast i utilització d'aquest protocol, així com un enriquiment del coneixement. D'altra banda, també es pretén progressar en l'àmbit de la programació i muntatge de sistemes i xarxes de controladors.

Paraules clau: CAN, emissions electromagnètiques i protocol de comunicació.

Resumen

El presente trabajo tiene como objetivo la construcción de un equipo didáctico que implemente una red de área de controladores basada en el protocolo BUS CAN (siglas del inglés *Controller Area Network*) sobre la que se pueda analizar su rendimiento en entornos industriales con altas emisiones electromagnéticas.

El sistema se ha implementado mediante placas de desarrollo de hardware, en este caso de Arduino®, controladores CAN comerciales y periféricos de entrada / salida que permiten gestionar las funciones básicas del equipo. La red CAN está constituida por dos nodos de transmisión de datos y el bus de comunicaciones estandarizado por el protocolo CAN. Cada uno de los nodos está formado por un microprocesador programable, un teclado y / o *display* y un módulo controlador / transceptor CAN que lo conecta al bus de comunicaciones. El *software* desarrollado en este trabajo facilita la gestión del sistema, las funciones de transmisión de datos y la visualización de los resultados de las comunicaciones a través de la red CAN.

Las pruebas de laboratorio realizadas en este trabajo han puesto de manifiesto la robustez del protocolo CAN y su idoneidad para utilizarlo en entornos industriales ruidosos con reducida carga de intercambio de datos entre los diferentes controladores y sensores del sistema.

Con la elaboración de este trabajo, se pretende formar una base sobre la que apoyarse a la hora de utilizar o de implementar proyectos con la tecnológica CAN, desglosando una competencia que presente soluciones y técnicas para la implementación, alcance y utilización de este protocolo, así como un enriquecimiento del conocimiento. Por otra parte, también se pretende progresar en el ámbito de la programación y montaje de sistemas y redes de controladores.

Palabras clave: CAN, emisiones electromagnéticas y protocolo de comunicación.

Abstract

The present project is focused in building a didactic equip that implements a Controller Area Network protocol in which its performance in industrial environments with high electromagnetic emissions can be analyzed.

The system has been implemented using hardware development boards, in this case from Arduino®, commercial CAN controllers in / out peripherals that allow the basic functions of the equip to be managed. The CAN network consists of two data transmission nodes and the communication bus standardized by the CAN protocol. Each node consists of a programmable microprocessor, a keyboard and / or a display and a CAN controller / transceiver module that connects it to the communication bus. The software developed in this project facilitates the system management, data transmission functions and the visualization of communications results through the CAN network.

Laboratory tests performed in this project have shown the robustness of the CAN protocol and its suitability for use in noisy industrial environments with a low data exchange load between the various controllers and sensors in the system.

With the elaboration of this project, it is intended to form a basis on which to lean when using or implementing projects with CAN technology, breaking down a competition that presents solutions and techniques for the implementation, scope and use of this protocol as well as enrichment of knowledge. On the other hand, progressing in the field of programming and assembly of controller systems and networks is also in this project's scope.

Keywords: CAN, electromagnetic emissions and communication protocol.

Agraïments

A la meva família,
per donar-me suport durant tots aquests anys de carrera i
sobre tot en aquest últim període del Treball de Fi de Grau.

Glossari

Recull de totes i cada una de les sigles emprades en aquest document.

ACK Acknowledgment.

AUI Attachment Unit Interface.

CAN Controller Area Network.

CiA CAN in Automation.

CRC Cyclic Redundancy Check.

CSMA/CD-BA Carrier Sense Multiple Access/ Collision Detection – Bit Arbitration.

DLC Data length Code.

ECU Electronic Control Unit.

EFLG Error Flag

EMC Electromagnetic Compatibility.

EMI Electromagnetic Interference.

EMS Electromagnetic Susceptibility.

EOF End of Frame.

FD Flexible Data-rate.

HLP High Level Protocol.

HS High Speed.

ICSP In-Circuit Serial Programming.

IDE Integrated Development Environment.

IDE Identifier Extension Flag.

IoT Internet of Things.

ISO International Organization for Standardization.

I2C Inter-integrated Circuit.

LAN Local Area Network.

LCD Liquid Crystal Display.

LLC Logical Link Control.

MAC Medium Access Control.

MAU Medium Attachment Unit.

MDI Media Dependent Interface.

MSB Most Significant Bit.

NRZ Non Return to Zero.

OSI Open System Interconnection.

PCS Physical Coding Sublayer.

PDU Protocol Data Unit.

PIC Programmable Integrated Circuit.

PMA Physical Media Attachment.

PWM Pulse Width Modulation.

REC Reception Error Counter.

RTR Remote Transmit Request.

SAE Society of Automotive Engineers.

SAN Storage Area Network.

SOF Start Of Frame.

SPI Serial Peripheral Interface.

SRR Substitute Remote Request.

TEC Transmission Error Counter.

UART Universal Asynchronous Receiver Transmitter.

USB Universal Serial Bus.

WAN Wide Area Network.

WiFi Wireless Fidelity.

Índex

RESUM I

RESUMEN II

ABSTRACT III

AGRAÏMENTS V

GLOSSARI VII

1. INTRODUCCIÓ 13

1.1. Origen del treball 13

1.2. Motivació 13

1.3. Requeriments previs 14

1.4. Objectius del treball 14

1.5. Abast del treball 14

2. ANTECEDENTS 15

2.1. Xarxa d'Àrea de Controlador (CAN) 15

2.2. Model OSI del sistema BUS CAN 17

2.2.1. Capa d'enllaç de dades del CAN 18

2.2.2. Capa física del CAN 20

2.3. Protocol de comunicació del CAN 22

2.3.1. Característiques generals 22

2.3.2. Transmissió *multicast* i filtratge de trames 22

2.3.3. Mecanisme de control d'accés al medi 23

2.3.4. Tipus de trames dels missatges CAN 23

2.3.5. Tipus d'errors, estats d'errors i mecanisme de confinament del protocol 27

2.4. Bus de comunicació de la CAN 30

2.5. Soroll en sistemes de potencia i de màquines elèctriques. Interferències radiades i conduïdes (EMI) 33

3. DESENVOLUPAMENT DEL PROJECTE 35

3.1. Disseny de l'escenari de treball 36

3.1.1. Descripció de l'equip 36

3.1.2. Instruccions d'ús de l'equip 37

3.2. Implementació *Hardware* 41
3.2.1. Disseny esquemàtic del projecte 42
3.2.2. Llistat de components 43
3.3. Implementació *Software* 45
3.3.1. Diagrames de flux 46
3.3.2. Definició de les variables emprades al *software.* 48
3.3.3. Explicació del software. 49
3.3.4. Llibreries utilitzades 50
3.4. Entorn i proves de laboratori 58
3.4.1. Objectiu de les proves de l'equip 58
3.4.2. Desenvolupament de l'objectiu 58
3.4.3. Alternatives per desafiar la robustesa de l'equip 61
3.4.4. Resultats de les proves 61

4. ANÀLISI DE L'IMPACTE AMBIENTAL 64

5. CONCLUSIONS 65

6. PRESSUPOST 66
6.1. Costs de materials 66
6.2. Costs de mà d'obra 66
6.3. Costs d'enginyeria 67
6.4. Cost total 67

BIBLIOGRAFIA 68

ANNEX A : CODI DEL PROGRAMA 71
A1. Codi del microcontrolador mestre 71
A2. Codi del microcontrolador esclau 76

ANNEX B: DATASHEETS DELS COMPONENTS 78

ANNEX C: ARXIUS .CPP I .H DE LES LLIBRERIES UTILITZADES 79

1. Introducció

Les xarxes d'àrea de controladors (CAN) són àmpliament utilitzades en la industria automobilística per gestionar les comunicacions internes de tots els controladors i sensors dels vehicles. El creixement del nombre d'automatismes i requeriments d'informació per part dels usuaris han fet créixer la importància d'aquest protocol i les seves característiques de funcionament.

En aquest treball s'aborda, d'una manera pràctica i amb una perspectiva didàctica, l'estudi i avaluació tècnica d'aquest protocol, construint un equip que implementa una xarxa de comunicacions utilitzant el protocol CAN i que permet mostrar-ne el seu funcionament i comportament en àmbits especialment hostils per a la seva fiabilitat.

1.1. Origen del treball

El treball va sorgir de l'idea d'unificar, en certa mesura, els camps de l'electrònica i les telecomunicacions, ja que són dos àmbits tecnològics en els quals sempre he estat interessat des del punt de vista de l'enginyeria. Es va comentar l'idea amb l'orientador i aquest va ser qui va proposar treballar amb un bus de comunicacions altament industrialitzat, programable i que requeria de muntatge electrònic: el BUS CAN.

1.2. Motivació

Les xarxes CAN són àmpliament utilitzades en sistemes de comunicacions de microcontroladors i sensors de molts entorns empresarials: automobilístic, industrial, aeroespacial, mèdic,... La majoria d'aquests entorns de treball, disposen de parts crítiques per la seva relació amb la seguretat o salut de les persones: activar l'airbag d'un cotxe, controlar l'escalfament dels motors d'un avió o monitoritzar les constants vitals d'un malalt. Moltes de les comunicacions associades amb aquestes tasques són realitzades a través del protocol CAN i per tant, cal realitzar estudis que analitzin la seva fiabilitat en front d'interferències.

Per altra banda, precisament derivat també de l'ampli ús actual a l'industria, és important disposar d'equips de laboratori que permetin l'estudi pràctic d'aquest tipus de protocol i xarxes de comunicació entre els estudiants d'electrònica.

1.3. Requeriments previs

Per poder realitzar una correcta lectura i comprensió del present document, es recomana al lector que sigui capaç d'entendre llenguatge de programació com pot ser C++, l'utilitzat per Arduino, així com disposar de coneixements d'electrònica i telecomunicacions.

1.4. Objectius del treball

Els objectius d'aquest treball es van fixar en dos àmbits. Per una banda es vol construir un equip que implementi una xarxa de comunicacions utilitzant el protocol CAN i que permeti estudiar el seu comportament a nivell de fiabilitat en entorns amb alt contingut d'interferències electromagnètiques. Per l'altra, l'equip ha de permetre mostrar el funcionament d'aquest tipus de sistema als estudiants d'electrònica per tal de facilitar-ne el seu estudi i comprensió des d'un punt de vista pràctic.

1.5. Abast del treball

Aquest treball de fi de grau inclou el disseny i construcció d'un equip que implementi un sistema de comunicacions basat en el protocol CAN i les proves de laboratori per a l'avaluació del seu comportament en entorns industrials amb altes components de radiacions electromagnètiques.

2. Antecedents

En aquest capítol es vol introduir els conceptes teòrics bàsics que s'han utilitzat en el treball i que es consideren necessaris conèixer per poder entendre el seu desenvolupament.

Entre aquests conceptes bàsics, s'han inclòs: una breu introducció a les xarxes de comunicació utilitzades en el treball, una descripció del protocol utilitzat per a l'intercanvi d'informació entre els nodes de la xarxa i fonaments del soroll generat per les interferències electromagnètiques de l'entorn on s'utilitza aquest tipus de xarxa.

2.1. Xarxa d'Àrea de Controlador (CAN)

Les xarxes de comunicació de dades, tradicionalment, s'han vingut classificant pel seu abast geogràfic o característiques funcionals. Així podem trobar, entre d'altres, les xarxes d'àrea amplia (WAN, Wide Area Network), xarxes d'àrea local (LAN, Local Area Network), xarxes d'emmagatzematge (SAN, Storage Area Network),... El treball que ens ocupa, utilitza una xarxa d'àrea de controlador coneguda per les seves sigles anglosaxones com CAN (Controller Area Network).

Des dels seus orígens al 1886 a Stuttgart, la companyia Robert Bosch GmbH es va dedicar al desenvolupament d'equips de mecànica de precisió i enginyeria elèctrica. Derivat dels seus treballs en l'àrea de l'automoció, al 1959 va començar a desenvolupar un sistema d'injecció de benzina controlat electrònicament. I a partir d'aquí, va començar a créixer el nombre de sistemes electrònics embarcats que controlaven els diferents elements de l'automòbil. Això va comportar que es treballés en desenvolupar un nou sistema de comunicacions per aquest nou entorn per poder deixar enrere la interconnexió punt a punt i la gran quantitat de cablatge associat. Al 1983, Robert Bosch GmbH va dissenyar un sistema de comunicacions específic que interconnectava els diferents nodes (controladors o sensors) del vehicle d'una forma eficient, flexible i amb alta fiabilitat. Aquells treballs van consolidar el terme de xarxes d'àrea de controlador (CAN) i van definir el sistema BUS CAN (bus i protocol CAN) de comunicacions que interconnectava físicament els nodes i gestionava l'intercanvi de missatges dins d'aquestes xarxes. Bosch va presentar oficialment el protocol CAN al 1986 en el congrés de la *Society of Automotive Engineers* (SAE) i un any més tard, al 1987, Intel treia al mercat el primer chip controlador de BUS CAN, el 82526.

L'especificació del BUS CAN desenvolupada per Bosch (versió 2.0) es va publicar finalment al novembre de 1993 sota la norma ISO 11898. Al 1995, aquest estàndard es va ampliar amb un apèndix que descrivia la extensió del format de les trames per incrementar el nombre de bits del camp d'identificador de node.

Les especificacions CAN revisades i actualitzades s'han normalitzat en els següents estàndards:

- ISO 11898-1 descriu la "capa d'enllaç de dades CAN i senyalització amb la capa física".
- ISO 11898-2 defineix la “unitat d’accés al medi d’alta velocitat de la capa física CAN".
- ISO 11898-3 especifica la "interfície de baixa velocitat, tolerant a les errades i depenent del medi de la capa física CAN".

Encara que originalment, el sistema BUS CAN d‘una xarxa d’àrea de controlador, va ser dissenyat específicament per a aplicacions al sector automobilístic, més tard, per la seva simplicitat, fiabilitat i immunitat electromagnètica, s’ha anat incorporant en aplicacions molt diverses: industrials, militars, aeroespacials, nàutiques, mèdiques, etc. Avui en dia, per exemple, gairebé tots els cotxes nous utilitzen aquest tipus de xarxa per gestionar els més de 70 ECU (*Electronic Control Unit*) que incorporen. I a curt termini, no sembla pas que hagi de canviar aquesta tendència. Ben al contrari, el sistema BUS CAN molt probablement encara serà més rellevant amb l’increment del nombre de controladors i sensors que cal interconnectar amb les noves tecnologies de *cloud computing*, l’Internet de les coses (IoT) i vehicles autònoms, que requeriran d’interconnexió i processament de la informació que rebran del mon exterior a través de xarxes mòbils. Actualment, ja es disposa de noves versions del protocol CAN, anomenat CAN FD (*Flexible Data-rate*), que permeten enviar més quantitat de dades en cada missatge (64 bytes) i incrementar la velocitat de transmissió màxima suportada (8 Mbit/s).

Les cinc avantatges més rellevants que han popularitzat aquest tipus de xarxa són ben conegudes: te un cost baix (utilitza una única interfície digital estandarditzat), permet el diagnosi i configuració centralitzada, és robust en front d’errors i interferències, eficient (els missatges són prioritzats i filtrats segons el node) i flexible (el nombre de nodes de la xarxa es pot modificar sense alterar el funcionament de la resta).

2.2. Model OSI del sistema BUS CAN

Tal com s'ha comentat a l'apartat anterior, les xarxes d'àrea de controlador utilitzen el protocol CAN estandarditzat a ISO per interconnectar els diferents nodes mitjançant un bus de comunicacions. Seguint el model de referència OSI, l'arquitectura del protocol CAN (ISO 11898) defineix les dues capes més baixes de la pila OSI de 7 capes: la capa d'enllaç de dades i la capa física.

El model OSI de capes, utilitza, bàsicament, els conceptes d'entitat de capa, primitives de intercanvi de serveis entre capes i unitats de dades per comunicar-se entre els diferents nodes d'una xarxa, com es mostra a continuació a la Figura 2.1.

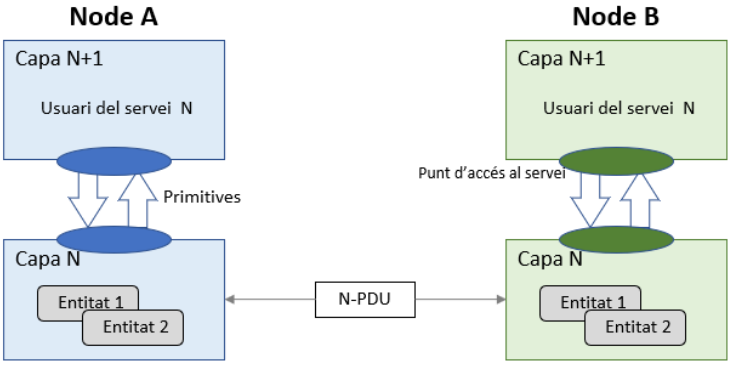


Figura 2.1. Model de capes OSI (Font: ISO [8])

D'aquesta manera, cada node d'un sistema BUS CAN implementa la pila de protocols definits a la ISO 11898. A cada capa es defineix una entitat de protocol que és l'encarregada de comunicar-se amb les capes immediatament superior i inferior mitjançant les corresponents primitives del protocol. I al mateix temps, les entitats "parelles" entre nodes, es comuniquen mitjançant trames o PDUs (*Protocol Data Unit*). Seguint les normes establertes pel protocol, qualsevol entitat d'una capa d'un node A pot comunicar-se amb la seva entitat parella d'un node B de la xarxa per intercanviar PDUs de capa que finalment donaran lloc a la interconnexió dels nodes per a la transferència de la informació requerida.

Com es representa a la Figura 2.2, la sèrie ISO 11898 especifica les regles per implementar les capes d'enllaç físic i de dades del model OSI per a la transferència d'informació entre dos o més dispositius de la xarxa.

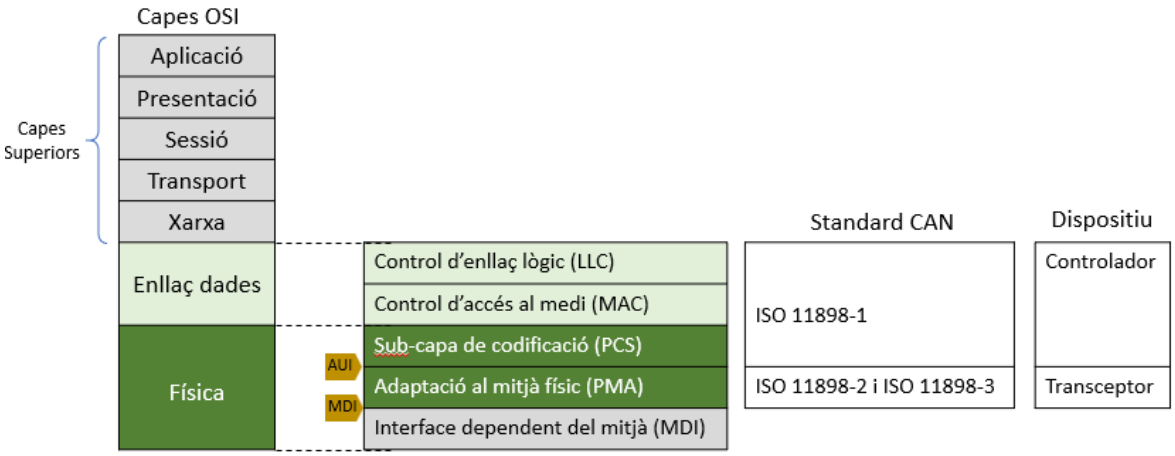


Figura 2.2. Model OSI implementat al sistema BUS CAN (Font: Microchip Technology Inc. [16])

Addicionalment, la industria ha desenvolupat capes superiors específiques per facilitar la integració en un únic dispositiu de les funcions bàsiques relacionades amb les capes inferiors del protocol CAN. El més genèric d'aquests protocols és el *CANopen*, desenvolupat per l'associació CiA.

2.2.1. Capa d'enllaç de dades del CAN

La capa d'enllaç de dades del protocol CAN implementa les funcions de control d'enllaç a nivell lògic (LLC) i la d'accés al medi (MAC) o bus.

2.2.1.1. Subcapa de control d'enllaç lògic (LLC)

La LLC ofereix dos tipus de serveis de transmissió no orientades a connexió: el servei de transferència de dades (punt a punt, *multicast* o *broadcast*) i el servei de sol·licitud de dades remotes a altres nodes. Aquests dos serveis donen lloc a dos dels tipus de trames definides pel protocol: trames de dades i trames remotes.

Les trames de dades LLC transporten dades des d'un node transmissor cap a un node receptor. Les trames remotes LLC s'utilitzen per sol·licitar la transmissió de trames de dades (amb el mateix identificador) des d'un node remot. En tots dos casos, la subcapa LLC també s'encarrega de notificar la transmissió o recepció correcta de la trama a l'entitat usuària.

Addicionalment als serveis esmentats, la subcapa LLC també proporciona les funcions de filtratge de trames acceptades, notificació de sobrecàrrega (*overload*) del dispositiu i de la gestió i tractament dels errors de la comunicació.

La primera de les funcions facilita el tractament de les trames rebudes pels buffers d'entrada mitjançant registres de filtratge que permeten descartar trames no desitjades pel node per tal de

lliurar-lo del seu processat. La segona funció permet alentir la recepció d'una nova trama quan el dispositiu està saturat de procés. La tercera, proporciona un mitjà automàtic de retransmissió d'aquelles trames que han perdut el mecanisme d'arbitratge establert pel protocol o s'han vist alterades durant la transmissió.

2.2.1.2. Subcapa de control d'accés al medi (MAC)

La subcapa MAC fa d'interfície entre la subcapa LLC i la capa física, atenent funcions i regles relacionades amb l'encapsulació/desencapsulació de dades transmeses/rebudes, detecció i senyalització d'errors i la gestió de l'accés al mitjà físic per a la transmissió/recepció de trames.

El model utilitzat per implementar la subcapa MAC es basa en la divisió en dos parts totalment independents, el procés de transmissió del de recepció. Això permet, entre altres coses, que un mateix node escoltant el que està transmeten, pugui detectar si ha generat algun error de codificació a la trama abans d'enviar-la al bus.

En el procés de transmissió, la subcapa MAC realitza l'encapsulació de la trama i la gestió d'accés al medi. Pel que fa a l'encapsulació, es validen les trames i interfície de control de la subcapa LLC, es calcula la seqüència CRC de la trama i es completa la trama segons el format que correspongui amb la resta de bits (SOF, SRR bit, IDE bit, RTR bit, reserved bits, CRC, ACK i EOF). Dins d'aquesta subcapa es gestiona l'accés al medi, que és una de les funcions més rellevants del protocol CAN: s'inicia la transmissió després de detectar el bus lliure, es serialitza la trama MAC, s'insereixen els bits de farciment (bit *stuffing*), es realitza el procés d'arbitratge per ocupar el bus passant a mode recepció en cas de perdre'l, es detecten errors, es revisa el bit d'ACK i la condició de sobre càrrega, es generen i envien les trames de sobre càrrega i de error i s'entrega el flux sèrie de bits a la capa física per a la seva transmissió.

En el procés de recepció de les trames, la subcapa MAC s'encarrega d'atendre la gestió d'accés al medi de la capa física i de desencapsular les trames segons la seva codificació. La gestió d'accés al medi comporta la recepció del flux de bits sèrie de la capa física, la desserialització i reconstrucció de la estructura de la trama, l'eliminació dels bits de farcit (bit de *stuffing*), la detecció d'errors (CRC, format o bit), la transmissió de l'ACK i la construcció i transmissió de les trames d'error i sobrecàrrega. Respecte al desencapsulat de les trames rebudes, aquí s'elimina la informació de la capa MAC de la trama rebuda i es trasllada la informació resultant a la subcapa LLC per al seu tractament.

La subcapa MAC està permanent supervisada per un mecanisme de gestió intern, anomenat "*Fault Confinement*", encarregat de delimitar els errors generats pels dispositius en fallida. Aquest és un mecanisme d'autocontrol del protocol per diferenciar els errors/fallides temporals dels permanents dels nodes i deixar fora del bus aquell dispositiu (node) que està afectat d'alguna fallida greu.

El procés de transmissió i recepció d'informació entre nodes d'un sistema BUS CAN es dur a terme i controla mitjançant quatre tipus diferents de trames: trames de dades, trames d'error, trames de sobre càrrega i trames remotes. Les trames s'envien al bus de comunicacions a través del transceptor que implementa la capa física del sistema.

2.2.2. Capa física del CAN

La capa física és l'encarregada de connectar un node CAN al bus de comunicacions. El nombre total de nodes CAN que podem incloure en el sistema està limitat per les càrregues elèctriques (impedàncies) del bus.

El model funcional definit per aquesta capa, representat a la Figura 2.2, la divideix en tres parts: la subcapa de codificació física (PCS) implementada en els xips del controlador CAN, la subcapa d'adaptació al mitjà físic (PMA) que implementa la circuiteria associada a les línies de transmissió/recepció del bus (transceptor) i la interfície d'accés al medi (MDI) que ofereix les característiques mecàniques i elèctriques requerides pels connectors que donen accés al medi físic que implementa el bus (cablejat).

Des del punt de vista de les capes OSI, la capa física s'encarrega de l'intercanvi de bits entre la seva entitat MAC local i les entitats corresponents de les subcapes MAC parelles dels altres nodes de la xarxa CAN.

Les funcions d'adaptació de la PMA junt amb la connexió de la MDI, s'agrupen en la MAU que es conegut habitualment a la industria com a transceptor CAN. La AUI és la interfície entre la subcapa PCS i la PMA on es defineix la interrelació entre el controlador CAN i el transceptor CAN del sistema BUS CAN.

Dins del sistema BUS CAN, el fabricant del sistema pot escollir qualsevol transceptor i mitjà de transport físic (bus) sempre que acompleixi els requeriments de la capa física CAN. Per això, la MAU (transceptor) s'implementa a partir de normes ISO específiques (ISO 11898-2 i ISO 11898-3) addicionals a la ISO 11898-1 (controlador CAN). La implementació de la MAU més habitual utilitzada a la industria és la ISO 11898-2 que defineix la MAU d'alta velocitat (HS-PMA), de fins a 1 Mbit/s per a distancies no superiors a 40 metres.

2.2.2.1. Subcapa de codificació física (PCS)

Dins d'aquesta subcapa es realitzen les funcions de codificació i descodificació dels bits al/del bus mitjançant la seva gestió durant el temps d'un bit de la trama. La lògica interna programable del temps de bit del controlador CAN facilita la sincronització dels nodes CAN, la compensació de retards de transmissió a la xarxa i l'establiment de la posició temporal del punt de mostreig de cada bit.

El punt de mostreig de bit marcarà el moment temporal precís en el que s'escriu o es llegeix el nivell del bus i s'interpreta com el valor "lògic" del respectiu bit.

La temporització del sistema es fixa a partir de la velocitat nominal del bit definida com el nombre de bits per segon transmesos per un transceptor ideal en absència de resincronització. El temps que dura un bit al bus el podem calcular com la inversa de la velocitat nominal de transmissió del bit.

Les freqüències dels oscil·ladors dels rellotges dels diferents nodes CAN s'han d'ajustar per proporcionar un quantum temporal específic que doni el mateix temps de bit per a tota la xarxa.

El protocol CAN contempla dos modes diferents de sincronització: la sincronització "*hard*" i la resincronització. Una sincronització "*hard*" força el reinici del temps de bit de tots els nodes de la xarxa mentre que una resincronització comportarà un escurçament o un augment del temps de bits de manera que la posició del punt de mostreig de bit sigui el correcte.

2.2.2.2. Subcapa d'adaptació al mitjà físic (PMA)

L'estàndard ISO 11898, com s'ha comentat, disposa de diferents normes per implementar la capa PMA. D'aquestes possibles implementacions, la més estesa i la que s'ha utilitzat en aquest treball és la ISO 11898-2, coneguda com HS-PMA.

La subcapa HS-PMA inclou una entitat transmissora i una entitat receptora. Aquesta subcapa s'encarrega de donar el nivell de voltatge adequat als suports físics connectats, normalment un cable elèctric de dos fils amb un terra comú. L'entitat emissora generarà sobre el bus un voltatge diferencial entre el senyal CAN_H i CAN_L per senyalitzar un 0 lògic (dominant) o no generarà aquest voltatge diferencial per senyalitzat un 1 lògic (recessiu). Aquests diferents nivells de voltatges són rebuts per la resta de nodes connectats a la mateixa xarxa CAN.

La subcapa HS-PMA disposa de dos interfícies amb les subcapes adjacents: l'AUI cap a la PCS i la MDI cap a la subcapa física depenent del mitjà. Ambdues interfícies disposen de les senyals TX, RX i GND. El senyal TX rep del PCS el flux de bits que ha de transmetre per la MDI. El senyal RX transmet cap a la PCS el flux de bits rebut per la MDI.

2.3. Protocol de comunicació del CAN

El funcionament del protocol CAN està especificat a l'estàndard ISO 11898-1 utilitzant el model OSI d'interconnexió de xarxes. El protocol estableix el funcionament de les dos capes més baixes del model deixant la implementació de la resta de capes a protocols d'alt nivell (HLP, *High Level Protocols*). En aquest capítol, a més de descriure les trames utilitzades pel protocol i el tractament dels errors, es vol fer èmfasi en explicar els mecanismes de gestió singulars que utilitza per confinar errors.

2.3.1. Característiques generals

El protocol CAN és un protocol de comunicació en sèrie, *half duplex* i asíncron que suporta el control i la transmissió *multicast* de forma distribuïda i en temps real.

El fet de que sigui un protocol sèrie implica que els bits s'envien un darrera l'altre pel bus de comunicació. El mode de transmissió que utilitza és *half dúplex* ja que tots els nodes de la xarxa poden transmetre i rebre, però no al mateix temps.

Podem dir que CAN és un protocol asíncron donat que no es transmet cap senyal de rellotge per sincronitzar els nodes, ni tampoc hi ha cap node mestre que tingui el rol de sincronitzar la resta de nodes i per tant, no hi ha un rellotge comú per a tots els nodes. En aquest protocol, la sincronització es realitza aprofitant el mètode de codificació de línia NRZ utilitzat per transmetre les dades i els flags de transició provocats per un canvi d'estat del bus, de recessiu a dominant o viceversa. Els nodes de la xarxa CAN estan sincronitzats per mostrejar el bus al mateix temps.

2.3.2. Transmissió *multicast* i filtratge de trames

A més, és un protocol de transmissió *multicast*, on tots els nodes poden emetre i rebre missatges de la resta de nodes del bus. Per això, el protocol disposa, al procés de recepció, de mecanismes de filtratge de missatges no desitjats per descartar aquells que no han de ser processats pel node.

Per implementar aquest filtratge, es important tenir en compte que el protocol CAN està orientat a missatge i no a node (adreça). Això vol dir que els missatges no s'envien d'un node a un altre basant-se en adreces, sinó que dins de cada trama CAN s'inclou la prioritat d'aquesta (basada amb el ID del node) i les dades a transmetre. Tots els nodes de la xarxa reben cada missatge transmès pel bus i confirmen (ACK) que s'ha rebut correctament.

Els nodes receptors són els encarregats d'acceptar o rebutjar la informació utilitzant el procés de filtratge d'acceptació de trames, que decideix si la informació rebuda és rellevant o no. Els receptors

no necessiten conèixer el transmissor de la informació ni viceversa. Aquest filtratge es configura als registres del controlador CAN destinats a aquestes tasques.

No obstant, malgrat és un protocol orientat a missatge, sí implementa la possibilitat de sol·licitar informació d'un altre node utilitzant trames del tipus "remota", que s'explica més endavant. En aquest tipus de missatges, en lloc de esperar a que un node enviï informació , és el mateix node el que sol·licita que el node remot li enviï la informació.

Un últim benefici d'aquesta característica del protocol basat en missatges és que els nodes no necessiten cap informació sobre la configuració del sistema (topologia, adreces dels nodes,...) per funcionar. I per tant, podem incloure o retirar qualsevol node de la xarxa sense necessitat de reprogramar o configurar cap dels altres nodes.

2.3.3. Mecanisme de control d'accés al medi

El protocol CAN implementa un mecanisme de control d'accés al bus del tipus CSMA/CD-BA (Carrier Sense Multiple Access / Collision Detection – Bit Arbitration). Aquest mecanisme regula la forma en què s'ha de produir la comunicació (escriure i llegir) dins de la xarxa amb un mitjà de transmissió compartit. L'arbitratge del BUS CAN garanteix que els missatges romandran intactes després de resoldre's inclús en el cas de detectar-se col·lisions i que per tant el missatge de més alta prioritat (ID més baix) no es veurà corromput ni retardat com a resultat d'aquest procés.

L'arbitratge s'inicia, si dos o més nodes del bus comencen les seves transmissions al mateix temps després de comprovar que el bus està en l'estat de repòs. Cada node envia els bits del seu missatge incloent el seu identificador de node i monitoritza l'estat del bus (recessiu o dominant). Mentre tots els bits enviats pels transmissors són idèntics, no passa res. Però en quan hi ha una diferencia, d'alguns dels bits, guanya l'arbitri el node que transmet un bit dominant i els nodes que han transmès un bit recessiu perden l'arbitri i canvien el seu mode de funcionament a "escoltar" transmeten bits recessius. Els nodes que perden l'arbitratge s'hauran d'esperar al següent període del bus en repòs per tornar a intentar transmetre els seu missatge.

2.3.4. Tipus de trames dels missatges CAN

La norma ISO 11898-1, dins de la subcapa MAC defineix el tipus de trames MAC que utilitza el protocol per intercanviar missatges dins de la xarxa.

El protocol CAN contempla quatre tipus diferents de trames: trama de dades, trama remota, trama d'error i trama de sobre càrrega. Les trames de tipus "dada" són utilitzades quan un node transmet informació a qualsevol o a tots els nodes de la xarxa. La trama "remota" és idèntica a la trama de dades però amb el bit RTR a '1' per indicar que és aquest tipus de trama. Les altres dos tipus de trames

serveixen per gestionar errors; la trama d'"error" és generada pels nodes que detecten qualsevol dels errors definits dins del protocol i la trama de "sobre carrega" es generada per nodes que requereixen més temps per processar missatges ja rebuts.

2.3.4.1. Trama de dades

Les trames de dades inclouen els camps d'arbitratge, control, dades, CRC, ACK i final de trama. El protocol CAN s'implementa en dos versions que es diferencien l'una de l'altra en la longitud de l'identificador de node (ID) que suporta (ID d'11 bits en la versió estàndard del protocol i ID de 29 bits en la versió estesa del protocol), tal i com es mostra en la Figura 2.3.

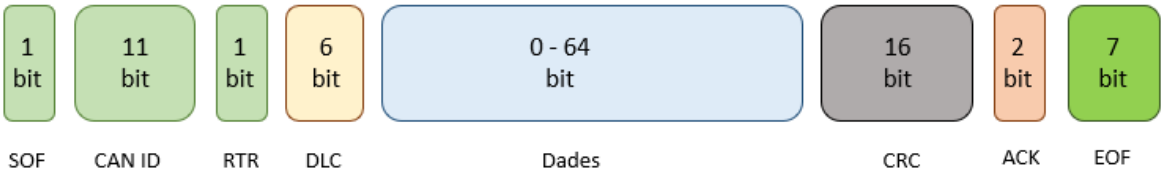


Figura 2.3. Codificació d'una trama de dades del protocol CAN (Font: ISO [8])

La trama s'inicia amb el bit SOF (*Start Of Frame*), que és sempre un bit dominant '0'.

El camp d'arbitratge s'utilitza per prioritzar missatges dins del bus com hem explicat anteriorment. Aquest camp té una longitud de 12 bits (11 bits d'ID i 1 bit d'RTR) o de 32 bits (29 bits d'ID, 1 bit que defineix el missatge de la versió estesa, 1 bit SRR sense us actualment i 1 bit RTR). El bit RTR a '1' permet identificar les trames del tipus remota.

Després del camp d'arbitratge segueix el camp de control on s'indica la longitud que tindrà el camp de dades. És un camp de 6 bits. El primer bit (MSB, *Most Significant Bit*) del camp és el bit IDE que ens permet diferenciar si una trama és de la versió estàndard (bit a '0') o estesa (bit a '1'). El següent bit és l'RB0 (i RB1 a la versió estesa) i està reservat. Els quatre bits restants del camp, DLC (*Data Length Code*) defineixen la longitud en bytes del camp de dades.

A continuació s'introdueix el camp de dades que inclou els bytes de dades que s'han indicat al camp de control. Aquest camp té una longitud variable de fins a 64 bits (8 bytes).

Seguidament, s'incorpora el camp CRC que consisteix en 15 bits de CRC i 1 bit delimitador de CRC. Aquest camp es utilitzat pels nodes receptors per determinar si s'ha produït algun error durant la transmissió. El polinomi generador del codi CRC i utilitzat pel transmissor i receptor és: $X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$. El node transmissor genera el codi CRC a partir del polinomi generador i insereix aquest camp a la trama. El node receptor realitza el càlcul amb el mateix polinomi sobre la trama

rebuda i compara el camp CRC rebut amb el càlcul realitzat. Si els dos codis coincideixen, la trama rebuda és correcta, però si és diferent, s'ha produït un error i caldrà retransmetre la trama.

A continuació del CRC, la trama disposa del camp de confirmació de trama rebuda correcta o ACK. En aquest *slot* de temps del bit ACK, el node transmissor escriu un bit recessiu al bus i escolta si algun receptor escriu un bit dominant '0' al bus. La resta de nodes, amb el resultat del càlcul del CRC, poden confirmar que la trama rebuda és correcta (bit '0') o que s'ha produït un error (bit '1') durant el temps de bit de l'*slot* ACK. Si algun receptor detecta un error, fica al bus un bit recessiu que provoca la finalització de la transmissió del missatge i la retransmissió del mateix per part del transmissor. Addicionalment, aquest procés ve acompanyat d'un mecanisme de confinament explicat més endavant per fer el tractament dels errors.

Per últim, la trama es finalitza amb el camp de fi de trama que consisteix en 7 bits de valor lògic '1'.

Després del final de la trama i tres bits de temps de repòs, anomenat "espai inter-trama", tots els nodes inclòs el que acaba de transmetre poden iniciar un nou procés de transmissió al bus.

A continuació, en la Figura 2.4 i Figura 2.5 es presenta un esquema simplificat de les trames que acabem de descriure:

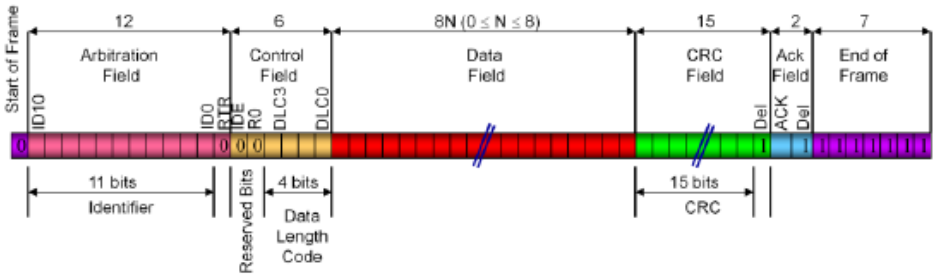


Figura 2.4. Estructura d'una trama de dades estàndard (ID amb 11 bits) (Font: Microchip Technolgy Inc. [12])

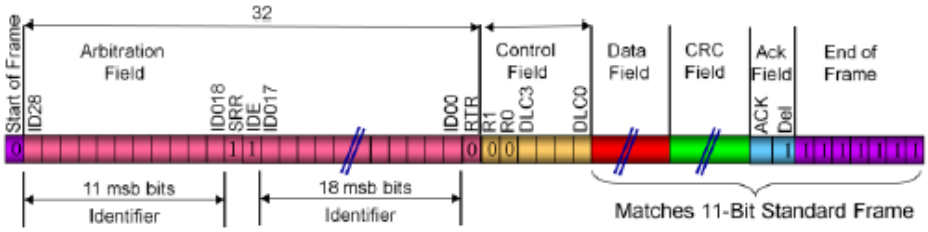


Figura 2.5. Estructura d'una trama de dades estesa (ID amb 29 bits) (Font: Microchip Technology Inc. [12])

2.3.4.2. Trama remota

Les trames remotes, són com les trames de dades però que no contenen camp de dades.

Les trames remotes s'utilitzen quan un node necessita sol·licitar alguna dada d'algun altre node. Un node que actua com a receptor de determinades dades pot iniciar la transmissió d'aquestes dades des de la seva font enviant una trama remota al bus.

Quan un node A (receptor) vol sol·licitar dades d'un node B (transmissor), transmetrà la petició mitjançant una trama remota on introdueix l'ID del node remot B del que vol aconseguir les dades i que coneix d'alguna trama de dades prèviament enviada pel node B al bus.

Després de llegir al bus el seu ID a la trama rebuda, el node B confirma amb el bit RTR a '1' que es tracta d'una trama remota dirigida a ell. El node B respondrà immediatament a la petició, transmeten una trama de dades amb la informació sol·licitada pel node A.

2.3.4.3. Trama d'error

Les trames d'error estan formades per dos camps diferents: el camp de *flag* d'errors i el camp delimitador d'error. El primer camp correspon a la superposició dels *flags* d'error dels diferents nodes del bus. El segon es simplement un delimitador d'aquest tipus de trama.

El camp de *flag* d'error pot utilitzar dos tipus de *flags* d'error, el *flag* d'error actiu i el *flag* d'error passiu. El *flag* d'error actiu consisteix en sis bits consecutius dominants '0' i el *flag* d'error passiu consisteix en sis bits consecutius recessius '1'. D'aquests últims, alguns o tots poden ser sobre escrits per altres nodes amb bits dominants.

Una trama d'error pot ser transmesa per qualsevol node quan aquest detecta un error amb una trama de dades o remota. En funció de l'estat d'error en el que es troba el node en aquell instant (actiu o passiu, descrit a l'apartat 2.3.5), transmetrà una trama d'error actiu o passiu, veure Figura 2.6. Qualsevol d'aquestes trames d'error provoca una violació de les regles marcades pels bits de farciment i destruirà la trama transmesa pel bus.

Figura 2.6. Estructura d'una trama d'error (Font: Microchip Technology Inc. [12])

Un node en estat d'error-actiu que detecti una condició d'error ho senyalitza enviant un *flag* d'error actiu. Com a conseqüència, tots els nodes detectaran la condició d'error i també transmetran un *flag* d'error. Per tant, la seqüència de bits dominants monitoritzats des del bus seran una superposició dels diferents *flags* d'errors enviats per la resta de nodes. La longitud total d'aquesta seqüència por variar entre 6 i 12 bits màxim. El mateix succeeix amb un node en estat d'error-passiu i flag d'error passiu.

Per altra banda, el camp delimitador d'error consisteix en 8 bits recessius.

2.3.4.4. Trama de sobre càrrega

El protocol CAN contempla la possibilitat de que es produeixi la condició de sobre càrrega des de la subcapa LLC o des de la subcapa MAC. Qualsevol d'aquestes condicions de sobre càrrega utilitza el mateix format de trama per avisar d'aquesta condició i retardar la transmissió al bus.

La trama de sobre càrrega també està composta per dos camps: el camp de *flag* de sobrecàrrega i el delimitador de sobre càrrega. El primer coincideix amb la mateixa estructura que el camp de *flag* d'error actiu i el segon amb el delimitador d'error.

L'enviament d'aquest tipus de trama al bus, provoca una petició de retard de transmissió de la pròxima trama a la resta de nodes i això permet al node més temps per processar el missatge actual.

2.3.5. Tipus d'errors, estats d'errors i mecanisme de confinament del protocol

El protocol CAN, tal com s'especifica a l'estàndard ISO 11898-1, defineix un llistat d'errors que es poden detectar durant la transmissió per assegurar la integritat dels missatges transmesos, provocant una retransmissió de la trama errònia.

Els errors detectats pel protocol poden ser dels següents tipus:

- Error de CRC.
- Error d'ACK.
- Error de format.
- Error de bit.
- Error de farciment.

Els errors detectats són coneguts per tots els nodes de la xarxa mitjançant les trames d'error o *flags* d'error. Qualsevol transmissió d'una trama errònia és tallada i la trama és retransmesa tan aviat com torna a guanyar el procés d'arbitratge del bus.

Per tractar de limitar els errors que es produeixen al bus i diferenciar errors esporàdics transitoris de condicions d'error més greus o permanents, el protocol CAN incorpora un mecanisme de confinament

de fallides (*Fault Confinement*) dels nodes basat en la utilització de tres estats d'errors en els que es pot trobar un node. Aquests estats segueixen una lògica de transició marcada per certes condicions dels comptadors d'errors de transmissió (TEC) i recepció (REC). A continuació, Figura 2.7 es presenta el diagrama d'estats de transició del protocol en funció dels valors d'aquests comptadors d'error:

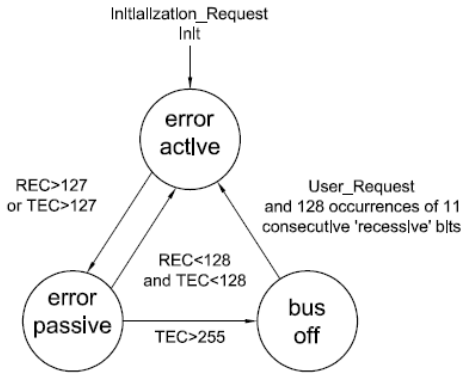


Figura 2.7. Diagrama de transició d'estats d'error d'un node (Font: ISO [8])

Tots els nodes s'inicialitzen a l'estat "*error active*", que és l'estat d'un node CAN durant l'activitat normal del bus. En aquest estat, el node pot transmetre qualsevol tipus de trama de les que disposa el protocol i hem comentat anteriorment.

Cada cop que en transmissió o recepció, el node detecta un error, incrementa els comptadors TEC i REC. Quan qualsevol d'aquests dos comptadors superar el valor de 127 errors, el node entra a l'estat "error passive". Estant en aquest estat, el node encara pot transmetre trames de dades, remotes i de sobre càrrega , però sols podrà transmetre trames d'error del tipus passiu. Com les trames d'error passiu són tots bits recessius '1', si el node continua detectant i transmeten trames d'error "passives", no provocaran que la resta de nodes de la xarxa incrementin els seus comptadors REC.

D'aquest estat d'error passiu, el node pot tornar a l'estat d'error actiu o passar a l'estat "*bus off*". Per tornar a l'estat actiu, els comptadors TEC i REC han de ser inferiors a 128. Els comptadors es decrementen cada cop que hi ha una trama vàlida al bus. Si per contra, el node continua incrementant el TEC per sobre dels 255, el node passarà a l'estat d'error "*bus off*". En aquest punt, el node es desconnecta del bus deixant de transmetre qualsevol tipus de trama.

Un cop el node ha entrat en aquest estat de "*bus off*", sols pot passar a l'estat "*error active*" mitjançant un canvi al mode de configuració del node. Aquest canvi del mode de funcionament, provoca la inicialització a zero dels seus comptadors TEC i REC.

Amb aquest mecanisme, el protocol aconsegueix confinar el node que provoca un elevat nombre d'errors de manera que l'impedeix operar i continuar incrementant els errors del bus. Això obliga a la revisió del node i tornar a entrar en el seu mode de configuració per a inicialitzar-lo.

2.4. Bus de comunicació de la CAN

El bus és el sistema de comunicació físic que interconnecta tots els nodes de la xarxa CAN.

A la Figura 2.8 es representa esquemàticament una xarxa CAN amb tres nodes interconnectats mitjançant un bus constituït per un parell de fils de coure carregats amb les corresponents resistències de 120 Ω per tal de delimitar el bus i evitar reflexions. Els transceptors es connecten al bus a traves dels seus punts de transmissió i recepció de senyals CAN_H i CAN_L.

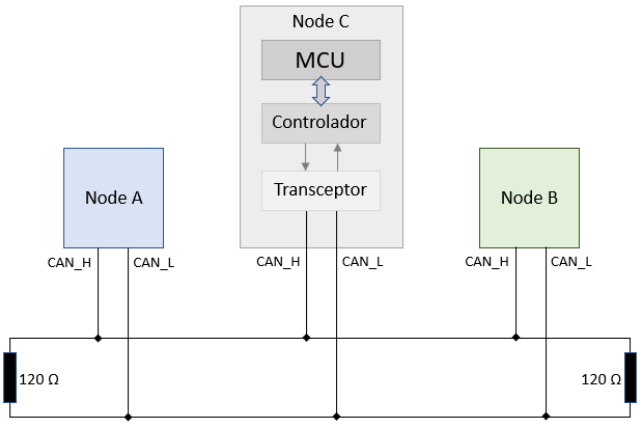


Figura 2.8. Exemple de bus de comunicació d'una xarxa CAN (Font: Microchip Technology Inc. [16])

Les característiques que defineixen el BUS CAN són els nivells de voltatge de senyal del bus i els connectors i cablejat utilitzats com a mitjà de transport.

El protocol CAN especifica dos estats lògics de treball del bus: recessiu i dominant. En cada instant, el bus està en un estat o en l'altre i els transceptors estan escoltant / llegint el bus o transmeten / escrivint pel bus. En la situació de repòs del bus, quan cap node està transmeten, es troba a l'estat recessiu. La ISO11898 defineix una diferencia de voltatge V_{DIFF} específica per a representar cadascun dels dos estats recessiu o dominant del bus: l'estat recessiu representa un bit lògic '1' amb $V_{CAN_H} = V_{CAN_L}$ i l'estat dominant representa un bit lògic '0' amb V_{CAN_H}-$V_{CAN_L} \neq 0$), tal i com es mostra a la Figura 2.9.

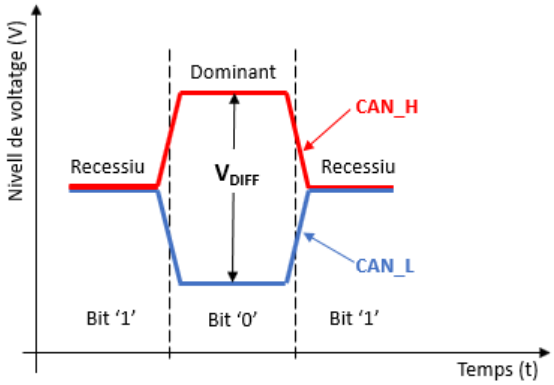


Figura 2.9. Definició d'estats lògics recessiu ('1') i dominant ('0') del bus CAN (Font: Microchip Technology Inc. [16])

A l'estat recessiu (lògic '1'), la diferencia de voltatge entre el port CAN_H i el port CAN_L és menor que el llindar mínim establert a la norma (<0,5V per a recepció o <1,5V per a transmissió).

Per contra, a l'estat dominant del bus (lògic '0'), la diferencia de voltatge entre el port CAN_H i el port CAN_L és més gran que el llindar mínim establert. Al bus, un bit dominant sempre sobreescriu un bit recessiu per tal d'implementar el mecanisme d'arbitratge de possessió del bus establert al protocol, tal i com es mostra en la Figura 2.10.

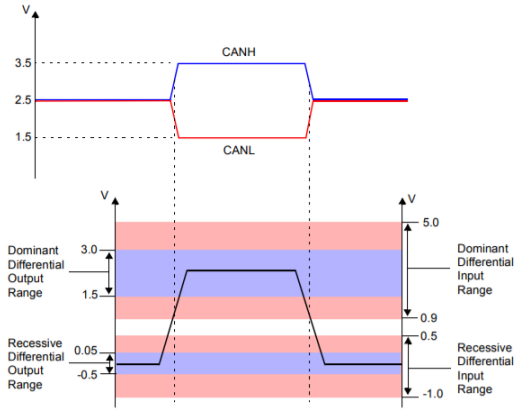


Figura 2.10. Nivells de diferencia de voltatge establerts pels estats recessiu i dominant del bus (Font: Microchip Technology Inc. [16])

Pel que fa als connectors i cables del bus, la norma ISO 11898-2 corresponent a la subcapa PMA, no especifica les característiques mecàniques dels cables i connectors. No obstant, si que requereix que aquests compleixin les especificacions elèctriques definides.

Addicionalment, la norma també requereix una terminació del bus amb una càrrega de 120Ω de resistència a cada extrem del bus.

La ISO 11898-2 especifica que qualsevol transceptor compatible ha de complir les especificacions elèctriques definides per tal d'aconseguir una alta robustesa elèctrica del bus. Algunes d'aquestes tenen per objectiu que el transceptor pugui suportar condicions elèctriques dures per tal d'assegurar la comunicació del node CAN. Així per exemple, el transceptor ha de sobreviure a curts circuits a les entrades del bus CAN de -3V a + 32V i a tensions transitòries de -150V a + 100V.

Per altra banda, la norma ISO 11898-2 especifica que els transceptors han de poder transmetre sobre un bus de fins a 40 m a la seva màxima velocitat (1 Mbit/s). Per aconseguir distancies més llargues del bus, s'ha de disminuir la velocitat de transmissió. No obstant, la màxima limitació de la longitud del bus la donarà el retard de propagació del transceptor utilitzat. Aquest retard es rellevant per implementar el mecanisme d'arbitratge d'utilització del bus, ja que cada node involucrat en el procés d'arbitratge ha de ser capaç de mostrejar cada bit dins del mateix temps de bit. Si dos nodes que lluiten per ocupar el bus es troben ens extrems oposats i el retard de propagació d'un d'ells es superior al de l'altre pot donar lloc a errors d'arbitratge per a lectures del bus fora del punt de mostreig.

Per últim, cal tenir present que hi ha solucions propietàries per implementar sistemes BUS CAN utilitzant altres mitjans d transmissió com ara fibra òptica o radio (WiFi). Aquestes solucions es basen en utilitzar adaptadors que converteixen els senyals de voltatge elèctric en senyals òptics o radio.

En el nostre treball el transceptor CAN que em utilitzat és el TJA1050 seguint l'especificació del fabricant Philips.

2.5. Soroll en sistemes de potencia i de màquines elèctriques. Interferències radiades i conduïdes (EMI)

Les interferències electromagnètiques (EMI) son aquelles pertorbacions que tenen lloc en circuits, components o sistemes elèctrics i que poden interrompre, degradar o limitar el rendiment d'aquests degudes o causades per una font de radiació electromagnètica externa o interna. N'existeixen de dos tipus: radiades i conduïdes i, habitualment, son generades en sistemes de potencia i de màquines elèctriques. Es consideren una de les fonts de soroll més important sobre els sistemes de comunicacions utilitzats per transportar informació entre els diferents nodes de la xarxa.

Les interferències conduïdes son aquelles que s'originen quan la senyal es propaga de la font a la víctima a través d'una connexió comú ambdues (per exemple la xarxa elèctrica). Les interferències radiades son aquelles que s'originen quan la propagació es per mitjà de radiació electromagnètica. Son precisament aquestes interferències radiades les interesants en aquest treball, ja que poden ser degut a camp elèctric i/o camp magnètic.

La presència de camps magnètics en convertidors commutats de potencia i màquines elèctriques es molt abundant i es una potent font de soroll per a sistemes de transmissió de informació. Aquest soroll dependrà en el cas de l'emissor, del flux magnètic (ϕ) present a la zona i de la seva variació ($d\phi/dt$) i en el cas del receptor, dependrà bàsicament del llaços conductors als bucles de transmissió que capten aquest flux i la seva variació.

Un dels aspectes més importants a tenir en compte pel disseny i implementació d'aquest tipus de sistemes, és la seva compatibilitat electromagnètica (EMC) mesurada en termes de la energia que la pròpia xarxa emet i la que es capaç de tolerar sense incorre en funcionament erroni (EMS, *Electomagnètic susceptibility*).

Donada l'alta densitat de components electrònics que avui en dia trobem en els sistemes de treball, la robustesa dels sistemes de comunicacions en vers aquest tipus de fonts de soroll és un element molt important a tenir en compte en el seu funcionament.

Els sistemes BUS CAN són molt robustos respecte aquests tipus d'interferències donat que aquestes afecten per igual al dos fils conductors del bus, mantenint la diferencia de voltatge que diferencia dos estats lògics diferents inalterable.

Tal i com es mostra en la Figura 2.11, degut a la naturalesa diferencial de la transmissió, la senyal CAN es poc sensible a les EMI, ja que ambdues línies son afectades per igual i l'amplificador diferencial del receptor les cancel·la.

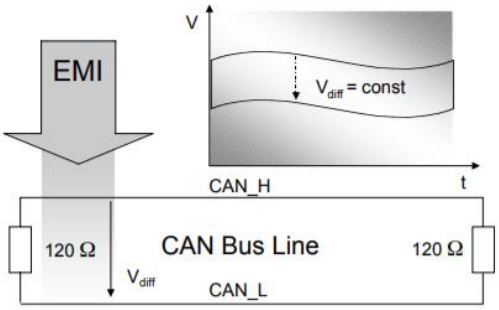


Figura 2.11. Efecte d'una EMI sobre les línies del bus CAN (Font: Martin Vázquez [17])

3. Desenvolupament del projecte

El treball s'ha elaborat en tres fases clarament diferenciades: una primera fase d'establiment de requeriments i objectius del projecte, una segona fase d'implementació (*hardware i software*) i una última fase de proves de laboratori.

A la primera fase s'estableixen els objectius del treball i els seus requeriments didàctics. El projecte ha de mostrar el funcionament de la tecnologia de comunicacions CAN i analitzar el seu comportament en entorns sorollosos. El sistema a implementar, per tant, ha d'esser un entorn de treball transportable que mostri el funcionament de la tecnologia CAN, àmpliament utilitzada en xarxes de interconnexió de controladors i sensors dins de sistemes industrials, i que permeti avaluar les seves prestacions en situacions d'estrès provocades per sorolls d'interferències electromagnètiques. Aquesta fase, també inclou adquirir els coneixements teòrics necessaris sobre la tecnologia CAN i realització del disseny funcional d'alt nivell del sistema.

Durant la fase segona es realitza la implementació *hardware* i *software* del sistema BUS CAN així com la seva adaptació als requeriments didàctics i de mobilitat establerts prèviament. La primera part de la implementació es centra en la construcció *hardware* i *software* del prototip; aquesta, bàsicament, consisteix en seleccionar els components utilitzats, procedir amb el muntatge i realitzar les proves bàsiques de funcionament. Un cop comprovat el correcte funcionament del prototip es desenvolupa el *software*, a partir de les llibreries disponibles, per implementar les funcionalitats completes requerides del sistema.

Finalment, durant la tercera fase es desenvolupen les proves de laboratori de l'equip sotmetent-lo a entorns de treballs amb altes components d'interferències electromagnètiques. Per aquestes proves s'ha utilitzat el laboratori d'electrònica de l'escola (UPC – EEBE).

En aquest capítol es procedeix a descriure amb més nivell de detall tan el disseny, com les implementacions *hardware* i *software* finalment realitzades.

3.1. Disseny de l'escenari de treball

L'escenari de treball a desenvolupar en aquest projecte ha de permetre mostrar el funcionament d'una xarxa CAN així com avaluar el seu comportament en entorns especialment sorollosos.

L'escenari de treball que s'ha implementat contempla la utilització de dos nodes d'una xarxa CAN interconnectats per un BUS CAN de comunicacions. El mitjà físic (cablejat) utilitzat per unir els dos nodes és modificable per tal de poder construir diferents entorns reals de funcionament de la xarxa.

A la següent figura es representa un esquema lògic i funcional del sistema implementat en el treball.

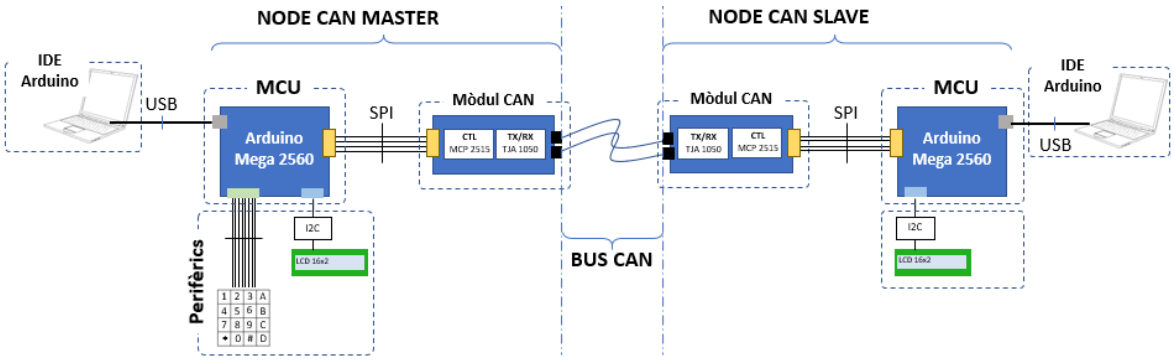


Figura 3.1. Esquema funcional del sistema CAN implementat

Com es pot observar, els dos nodes incorporen una placa amb microprocessador (MCU) i una placa BUS CAN (controlador CAN i transceptor CAN). El microprocessador del node mestre incorpora el software de gestió funcional de l'equip i implementa les comunicacions CAN amb el node esclau. Per la seva banda, el microprocessador del node esclau implementa les comunicacions amb el node mestre. Els nodes utilitzen perifèrics (teclat i *displays* LCD) per introduir dades i mostrar els resultats de les comunicacions realitzades.

3.1.1. Descripció de l'equip

L'equip al que aquest projecte fa referencia, permet a l'usuari, comprovar el funcionament del BUS CAN i obtenir diferents dades sobre els errors ocorreguts durant les transmissions de missatges. Aquest equip permet, per mitjà d'un menú d'opcions, realitzar transmissions d'una durada de 10 segons (o mes curta en cas de pausar-la) a una velocitat de 1.320 missatges per segon i seguint un patró prefixat. Aquestes transmissions permeten comprovar l'estat, a nivell d'error, en el que es troba el bus CAN; depenent del valor dels comptadors d'error del controlador CAN MCP2515, el bus es trobarà en un dels següents estats, esmentats anteriorment:

Error Active: Si el valor dels comptadors d'errors de transmissió o recepció tenen un valor menor a 127. És l'estat de funcionament normal i en el que el node pot transmetre missatges de dades, remots, sobre càrrega i d'error del tipus error actiu (formades per bits dominants) sense cap restricció.

Error Passive: Si el valor dels comptador d'errors de transmissió o recepció tenen un valor superior a 127. En aquest estat es poden continuar transmeten missatges de dades, remota i sobre càrrega, però les trames d'error transmeses són del tipus passiu (formades per bits recessius).

Bus off: Si el valor del comptador d'errors de transmissió te un valor superior a 255. Aquest estat fa que aquella estació quedi temporalment inhibida de participar sobre el bus de comunicacions. Mentre duri aquest estat, el node afectat no pot ni enviar ni rebre missatges. Només els transmissors poden adquirir l'estat de *Bus off.*

D'aquesta manera, l'usuari pot comprovar el correcte funcionament del protocol de comunicacions bus CAN en determinades circumstancies com per exemple un lloc amb alta radiació electromagnètica.

3.1.2. Instruccions d'ús de l'equip

L'usuari pot interactuar amb l'equip mitjançant els perifèrics disposats per aquesta finalitat. El node CAN mestre disposa d'un teclat i un display LCD mentre que el node CAN esclau sols disposa d'un display LCD informatiu.

Un cop alimentat, l'equip dona la benvinguda i instants després mostra un menú d'opcions en la pantalla del dispositiu mestre i un missatge d'espera a la pantalla del dispositiu esclau, tal com es mostra a continuació:

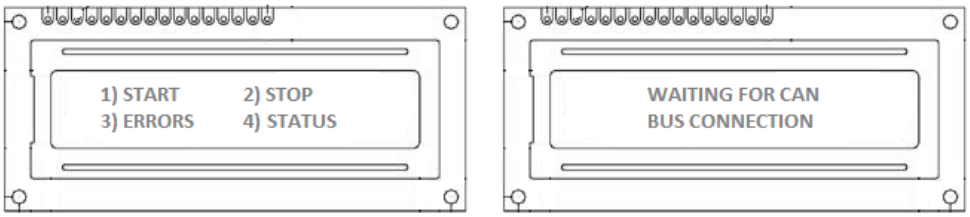


Figura 3.2. Menú principal de l'equip

Aquesta es la interfície d'interacció principal de l'equip i a partir de la qual, en funció de la tecla que es pressioni del teclat, es redirigirà a la opció escollida.

La **tecla (1)** desenvolupa la funció primària del dispositiu, en pitjar-la l'equip inicia la transmissió de dades pel bus CAN. Tal i com s'ha explicat prèviament, aquesta transmissió de trames de dades durarà 10 segons i l'usuari visualitzarà els següents missatges informatius:

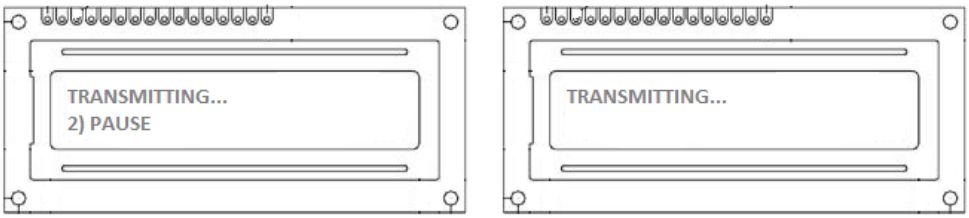


Figura 3.3. Aspecte dels LCDs durant la transmissió

Per una banda l'LCD del dispositiu esclau mostra la paraula "transmetent" parpellejant indicant que la transmissió està activa. En el moment que la transmissió finalitzi o es pausi, deixarà de parpellejar. D'altra banda l'LCD del dispositiu mestre suggereix pressionar la tecla (2) per pausar la transmissió.

Al ser pressionada la **tecla (2)**, el dispositiu presenta els següents missatges a l'usuari i pausa la transmissió de dades:

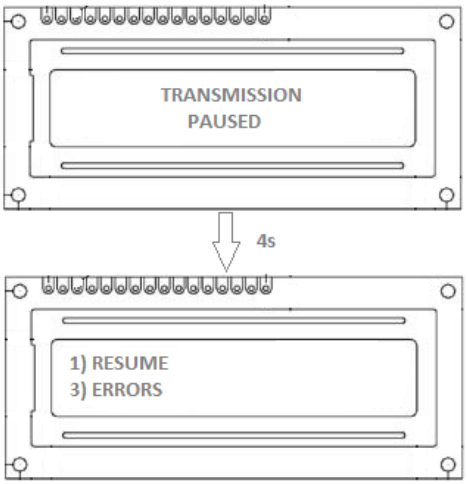


Figura 3.4. Estat de l'LCD master en pausar la transmissió

A cap de 4 segons, el dispositiu suggereix dues opcions. La primera, pressionant la tecla (1) per segon cop, ens retorna a l'estat de la Figura 3.3. Es tracta de l'opció de continuar per l'instant on s'havia deixat la transmissió abans de ser pausada. Aquesta opció només te sentit quan es l'usuari qui interromp la transmissió manualment. La segona, pressionant la **tecla (3)**, condueix al recull d'informació d'errors ocorreguts durant la transmissió, mostrant per la pantalla del node esclau el següents missatges informatius:

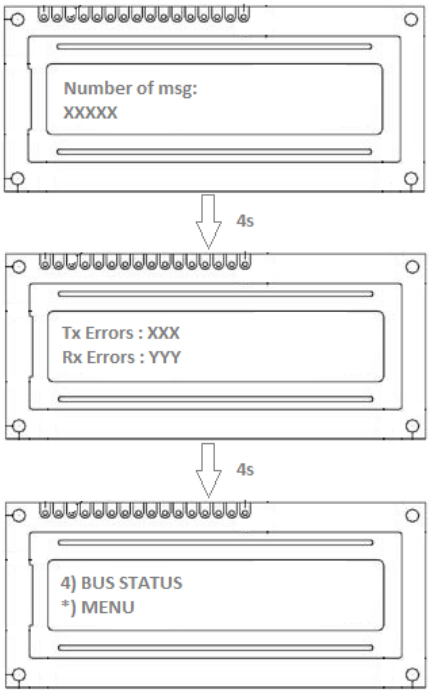


Figura 3.5. Estat del LCD *slave* al accedir a la interfície d'errors

En primer lloc, el dispositiu retorna el nombre de missatges enviats durant la transmissió, passat un breu instant de temps, apareix el nombre d'errors de transmissió (Tx) i recepció (Rx) i finalment suggereix dues noves opcions (4) o (*). Al pressionar la **tecla (4)** es mostra la informació sobre l'estat d'error en el que es troba el bus CAN, que tal i com s'ha explicat al inici d'aquest apartat, pot variar depenent del nombre d'errors de transmissió i recepció entre: *Error active, Error passive* i *Bus off.* Al prémer aquesta tecla, apareixerà per l'LCD el següent:

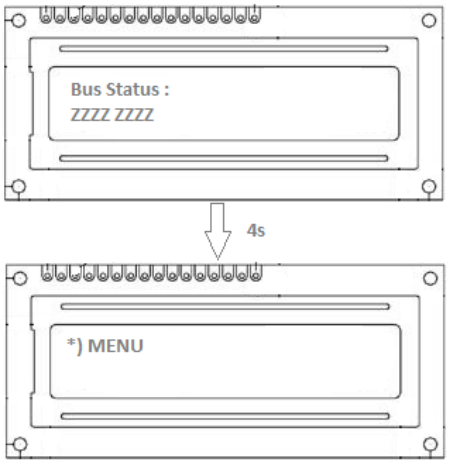


Figura 3.6. Aspecte de l'LCD *master* en presentar l'estat actual del node CAN

Transcorregut un cert instant de temps, el dispositiu mostrarà únicament l'opció de la **tecla (*)** la qual realitza un RESET en tots els camps i retorna al menú principal de la Figura 3.2**.** Això vol dir que al pressionar aquesta tecla es pot realitzar una nova transmissió de dades, esborrant així la informació (missatges enviats, errors de transmissió i estat del bus) que teníem fins ara.

3.2. Implementació *Hardware*

L'equip del treball s'ha construït a partir de l'especificació de requeriments d'aquest TFG i seguint el disseny realitzat prèviament. La construcció de l'equip s'ha realitzat en dos fases: una primera fase de prototip que ha servit de punt de proves i validació funcional de requeriments i una segona fase d'adequació i mecanització d'aquest per aconseguir un equip de treball didàctic portable.

El prototip ha servit per establir les bases de coneixement dels diferents elements que composen els sistema i realitzar les proves de funcionament dels mòduls software desenvolupats. Durant aquesta fase s'han analitzat els *datasheet* dels components utilitzats per interconnectar-los apropiadament i poder utilitzar els *pins* de referencia que ha d'utilitzar la capa de software. Precisament, una part important d'aquesta fase ha estat la integració del hardware amb el software que ha obligat a revisar el cablejat i *pins* de connexió fins a aconseguir el funcionament desitjat.

Un cop validada l'operativa i funcionalitat del prototip i realitzades les proves de laboratori s'ha dut a terme una adaptació mecànica per convertir l'equip en un instrument de treball didàctic fàcilment transportable i accessible per futures manipulacions dins de l'àmbit de l'ensenyament. Amb aquesta finalitat s'han integrat tots els elements en una caixa de metacrilat transparent que permet visualitzar els diferents components i facilita la seva manipulació tan per modificar el software utilitzat com els cables del bus de comunicacions dels dos nodes CAN.

El resultat d'aquesta implementació es mostra a la següent imatge.

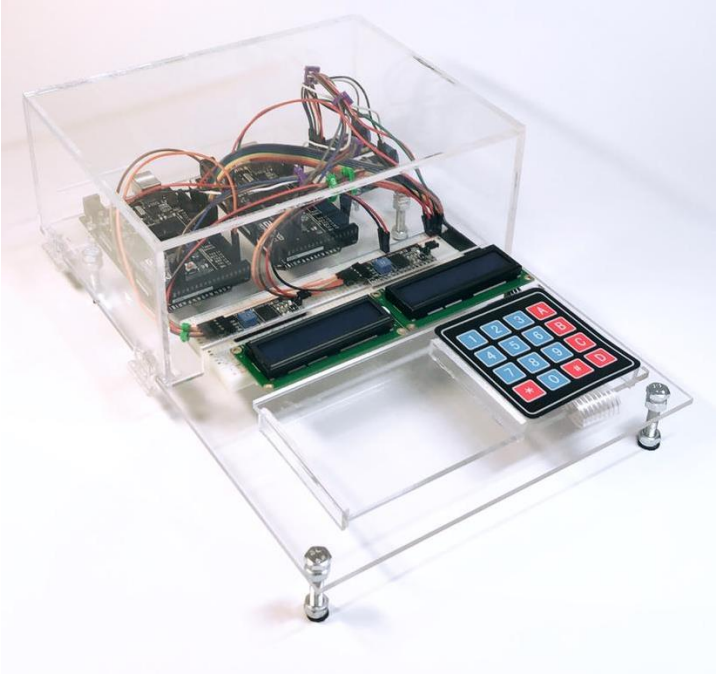
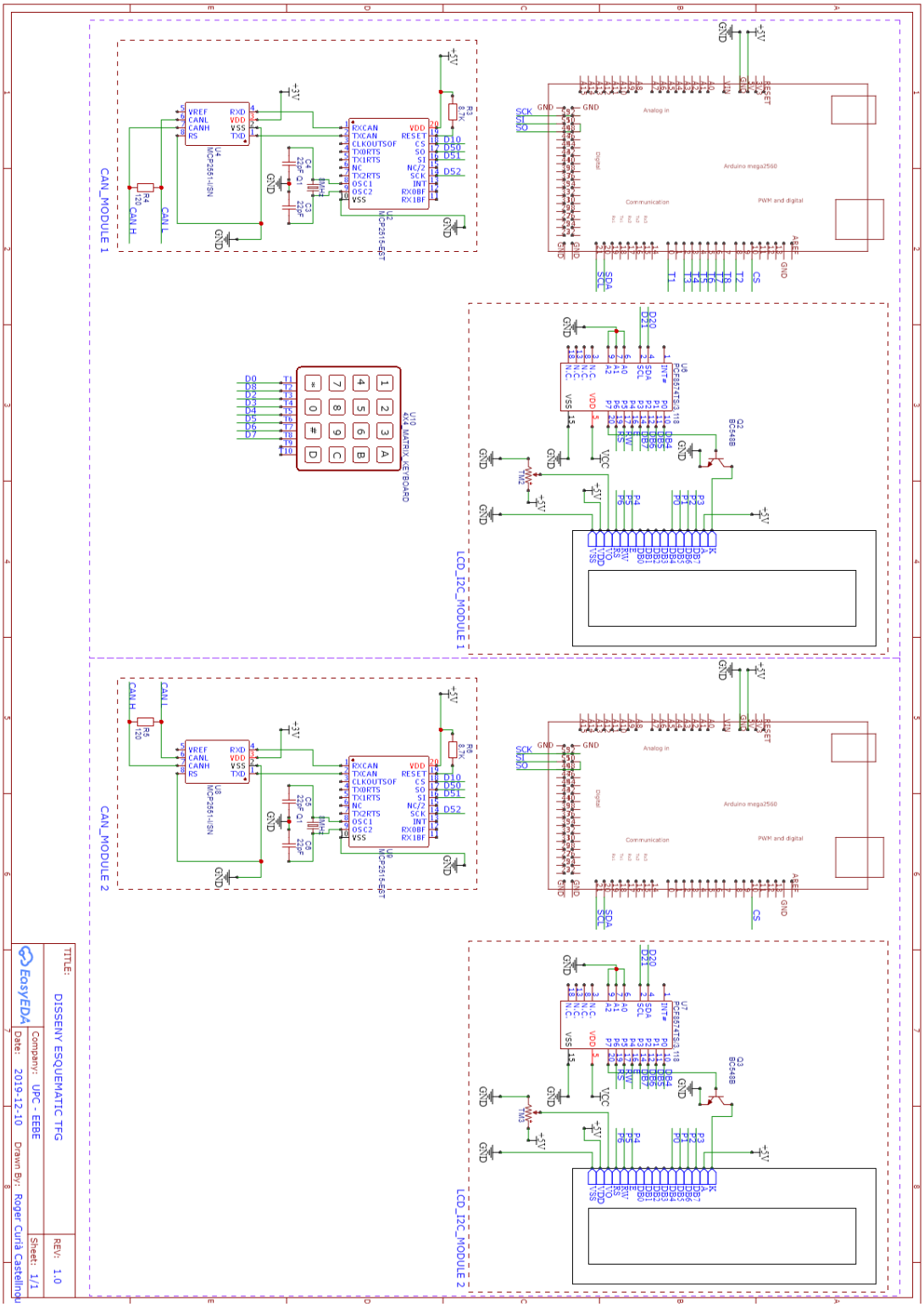


Figura 3.7. Imatge de l'equip de treball construït en aquest projecte.

3.2.1. Disseny esquemàtic del projecte



Per una millor visualització d'aquest disseny esquemàtic, **descarregui i executi** l'arxiu que es troba al següent enllaç directe: https://drive.google.com/open?id=1bMXutLl6CYmUk6B9eysbgSdnLmA8ETY3

3.2.2. Llistat de components

En aquest apartat es llistaran els components emprats al projecte.

COMPONENTS ELECTRÒNICS:

1. **Arduino Mega 2560**: és una placa microcontroladora basada en l'ATmega2560. Disposa de 54 pins d'entrada / sortida digitals (dels quals 15 poden utilitzar-se com a sortides PWM), 16 entrades analògiques, 4 UARTs (ports sèrie *hardware*), un oscil·lador de cristall de 16 MHz, una connexió USB, una presa de corrent, una capçalera ICSP, i un botó de *reset*. *Datasheet* disponible a la referencia **[1]** de l'annex B.

2. Pantalla de Cristall Líquid 16 x 2: l'LCD (de l'angles *Liquid Crystal Display*) es una pantalla formada per un conjunt de píxels posicionats al davant d'una font de llum que permet mostrar una sèrie de nombres i caràcters en ser programat. En aquest cas l'abast màxim de caràcters o nombres a mostrar en un instant es de 32 (16 per fila). *Datasheet* disponible a la referencia **[2]** de l'annex B.

3. Mòdul CAN-BUS: Mòdul que permet connectar un microcontrolador com pot ser una placa Arduino al bus de comunicacions CAN. Aquest mòdul esta format per un controlador MCP2515 amb interfície SPI i un transceptor TJA1050. Per tal d'establir una comunicació mínima son necessaris 2 d'aquests mòduls. *Datasheet*s disponibles a la referencia **[3]** de l'annex B.

4. Mòdul adaptador LCD a I2C: Mòdul que permet manipular el *display* utilitzat únicament amb 2 pins (SDA i SCL). Aquest mòdul es basa en el controlador PCF8574. Les adreces I2C per defecte del mòdul poden ser 0x3F o 0x27 (aquesta última, es el nostre cas). Cal identificar correctament en quina posició es troba el mòdul per tal de que el programa funcioni correctament; per a tal, es pot compilar un petit *sketch* d'Arduino que s'anomena "I2C *Scanner*" que en pocs segons ens mostrarà pel port serial el valor de la nostra posició. *Datasheet* disponible a la referencia **[4]** de l'annex B.

5. Teclat matricial 4 x 4: Aquest teclat disposa de 4 botons per cada una de les 4 files. Es un teclat de membrana on les files i les columnes estan connectades d'una forma determinada. Les files marcaran (d'esquerra a dreta) els 4 primers pins, mentre que les columnes seran els 4 últims. De manera que en pressionar per exemple la tecla numero 1, el polsador tancarà el circuit que existeix entre el pin 1 (fila 1) i el 5 (columna 1). *Datasheet* disponible a la referencia **[5]** de l'annex B.

6. Placa d'inserció o *Protoboard*: Es tracta d'una placa amb orificis que estan connectats entre ells internament amb la fi de poder inserir components electrònics per tal de realitzar petits circuits elèctrics d'una manera molt flexible.

7. Cables *dupont*: Cables que actuen de pont entre dos components. N'existeixen de diferents longituds així com també pot variar la seva connectorització (femella-femella, mascle-mascle o mascle-femella).

8. Cable USB – Arduino: Es tracta d'un cable amb una de les terminacions USB A (la que s'endolla al PC) i l'altra USB B (la que s'endolla a l'Arduino). Per tal d'alimentar la placa microcontroladora a demés de poder carregar els corresponents programes en ella.

9. Connectors de fila de pins o connector Berg: Connector elèctric format per una o mes fileres de pins mascle un a continuació de l'altre.

COMPONENTS MECÀNICS:

1. Planxa de metacrilat: Lamina de 0.4 cm de gruix de metacrilat transparent, diferents mesures.

2. Frontissa.

3. Cargols i femelles.

4. Ventoses anti-lliscament.

3.3. Implementació *Software*

La funcionalitat de treball que ofereix l'equip s'aconsegueix amb un *software* que implementa les funcions de transmissió (emissió i recepció) als nodes CAN del sistema i informa de l'estat de certes variables i registres accessibles dels controladors.

La programació s'ha desenvolupat amb la plataforma IDE de software lliure que ofereix Arduino i que permet carregar pel bus USB als microprocessadors MEGA 2560 que s'han utilitzat al treball com elements intel·ligents que gestionen les transmissions dins del sistema CAN.

El *software* desenvolupat disposa de diverses rutines per implementar les diferents funcionalitats de l'equip: menú, transmissió, recepció i presentació de resultats.

A continuació es descriuen els diagrames de flux que representen el funcionament de l'equip.

3.3.1. Diagrames de flux

3.3.1.1. Diagrama de flux del *Software*

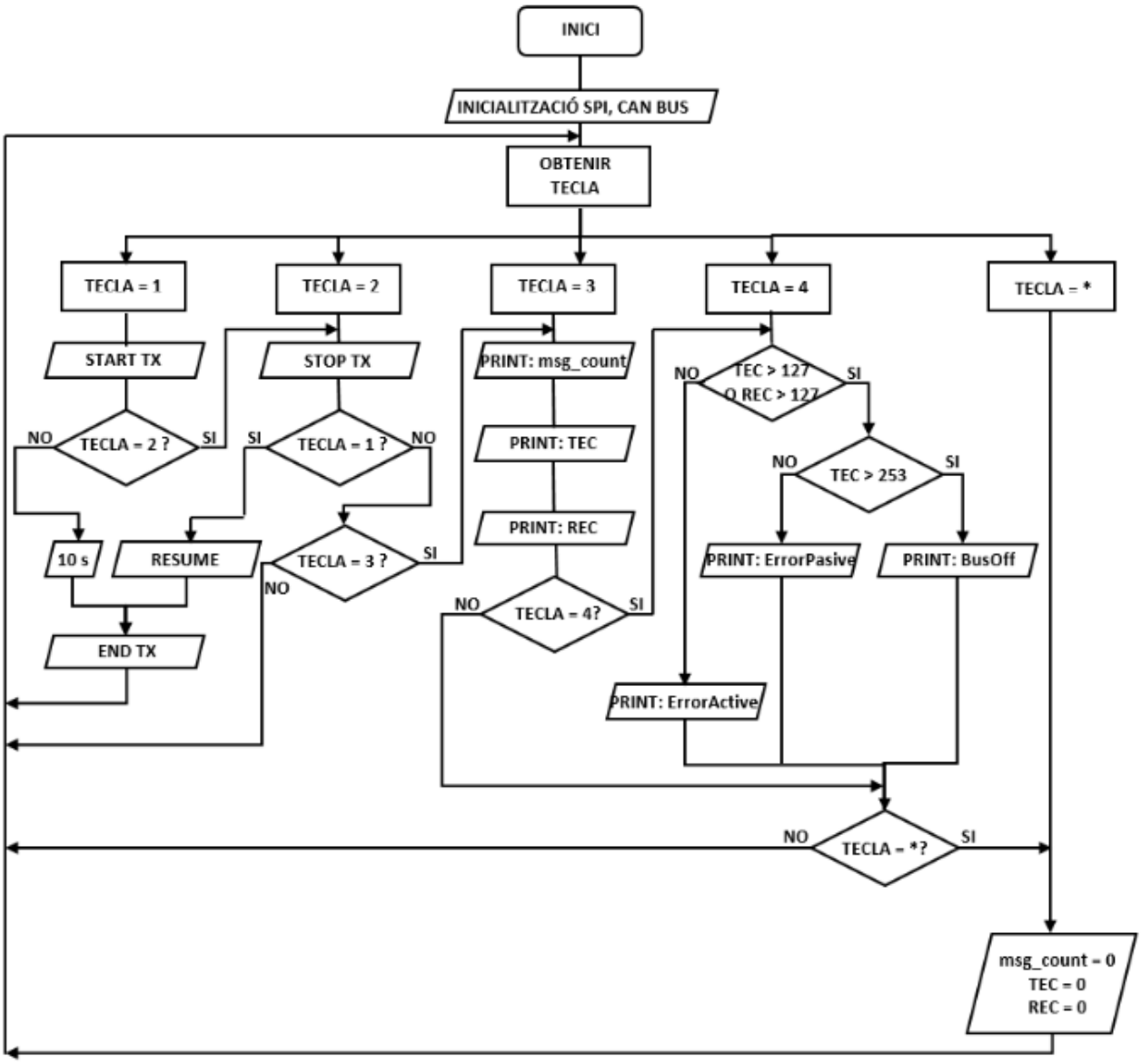


Figura 3.8. Diagrama de flux del funcionament principal del projecte

3.3.1.2. Diagrama de flux del procés de transmissió de dades

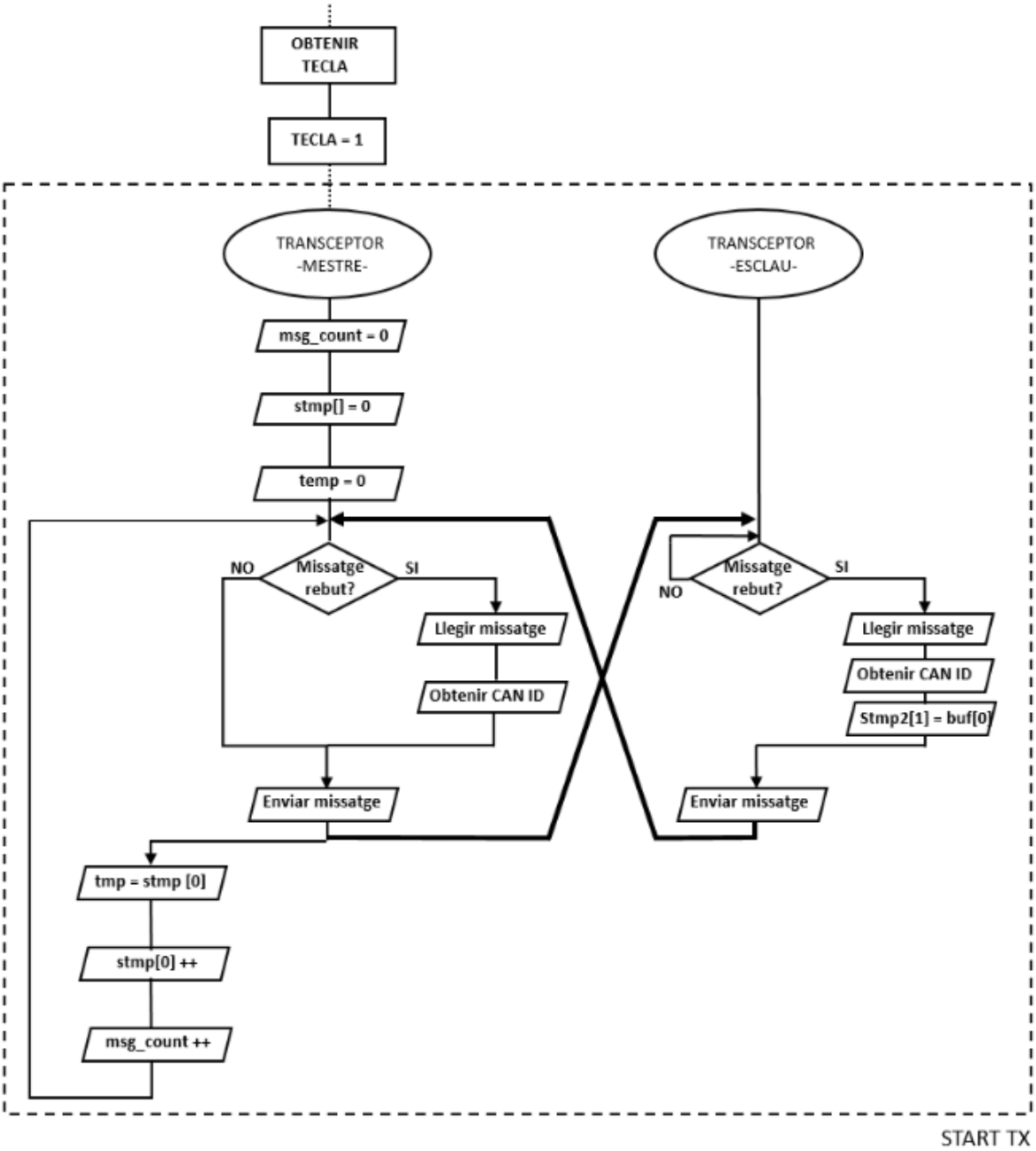


Figura 3.9. Diagrama de flux del procés de transmissió al polsar la tecla 1 (*Start*)

3.3.2. Definició de les variables emprades al *software.*

En aquest apartat i per tal de facilitar la comprensió del lector, es recolliran les variables utilitzades en la programació així com la seva breu definició corresponent.

- CODI TRANSCEPTOR (*MASTER)***:**
  1. **SPI_CS_PIN:** Variable en la qual fixarem el valor del pin digital de l'Arduino on col·locarem el cable de CS (*Chip Select*). De manera que si en el futur es canvia el microcontrolador, es pugui adaptar aquest valor fàcilment.
  2. **msg_count**: Variable inicialment a zero que incrementarà el seu valor en una unitat cada vegada que s'envia un missatge pel bus.
  3. **stmp:** *Buffer on s'emmagatzemarà* el valor del camp de dada del missatge a enviar. Inicialment a zero, aquest valor s'incrementarà en una unitat fins arribar a 255, quan retornarà a 0.
  4. **temp:** Variable temporal la qual emmagatzemarà el valor del camp de dada que s'ha enviat pel bus, es a dir la primera posició del *buffer* **stmp**. La utilitat d'aquesta variable es poder guardar el valor que presenta **stmp** abans de ser enviat per tal de poder comparar-la amb el retorn de tal valor, de manera que ambos valors quedin sincronitzats, tal i com es mostra en la Figura 3.10.

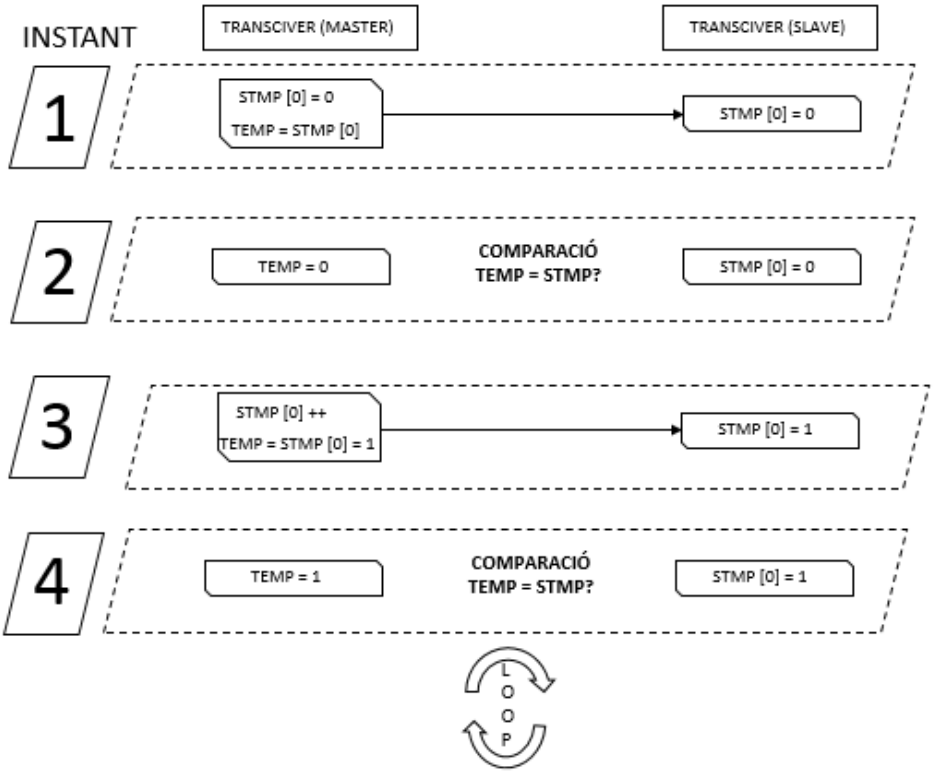


Figura 3.10. Explicació de la funcionalitat de la variable temporal "temp"

5. **len2:** Aquesta variable determina el valor, en bytes, de la longitud del camp de dades rebut. Inicialment es troba a 0.
6. **buf2:** *Buffer* on s'emmagatzemarà el contingut del camp de dades rebut.
7. **CErrorTEC, CErrorREC, CErrorEFLG**: Variables on guardarem el valor dels registres d'errors TEC (*Transmit Error Counter*), REC (*Receive Error Counter)* i EFLG (*Error Flag*).

- CODI TRANSCEPTOR (*SLAVE)*:
  8. **len:** Aquesta variable determina el valor, en bytes, de la longitud del camp de dades rebut. Inicialment es troba a 0.
  9. **buf:** *Buffer* on s'emmagatzemarà el contingut del camp de dades rebut.
  10. **stmp2**: *Buffer on s'emmagatzemarà* el valor del camp de dades del missatge a enviar. El seu valor dependrà del valor que presenti **buf** en cada instant.

3.3.3. Explicació del software.

El codi que s'adjunta a l'apartat d'annexos del projecte es pot estructurar en dues parts:

1. CONTROL DE DISPLAYS I TECLAT (MENÚ D'OPCIONS)

Mitjançant l'ús d'estructures condicionals, essent així la condició: pressionar una tecla del teclat matricial, podem realitzar un conjunt d'opcions que mostrarem per pantalla (START, PAUSE, ERRORS, STATUS, RESET). Cada una d'aquestes opcions correspon a la condició de pressionar una tecla (1, 2, 3, 4, *).

2. TRANSMISSIÓ (EMISIÓ/RECEPCIÓ)

Per tal de realitzar una correcta transmissió necessitem dos nodes a la xarxa (estacions). La primera s'encarregarà de generar i enviar els missatges i rebre'ls de tornada (Mestre), i la segona es dedicarà a rebre el missatge enviat pel primer i immediatament reenviar-se'l (Esclau). Fent servir les funcions que conté la llibreria del BUS CAN que més endavant es presenta, implementarem el *software* de les dues estacions.

El *software* es basa en un bucle que està corrent durant 10 segons. Dins del bucle, l'estació mestra no pararà d'enviar trames amb el valor de **stmp** en el camp de dades. Paral·lelament, l'estació esclava rebrà les trames amb el contingut de **stmp**, com a **buf** i l'emmagatzemarà a **stmp2.** A continuació iniciarà el procés de retransmissió, enviant el contingut del valor del **buf** dins d'una trama amb el seu identificador (ID) pel bus cap a l'estació mestra. Addicionalment, el *software* permet aturar temporalment el procés de transmissió, prement la tecla 2 abans de finalitzar el període de 10 segons i reinicialitzar-lo tornant a prémer la tecla 1 (RESUME).

3.3.4. Llibreries utilitzades

3.3.4.1. Teclat matricial (KEYPAD.h)

Descripció:

Es tracta d'una llibreria externa a les de per defecte d'Arduino però acceptada per l'empresa. Creada per Mark Stanley i Alexander Brevig, aquesta llibreria permet la utilització de teclats matricials amb Arduino. No es necessari l'ús de resistències ni de díodes externs ja que la mateixa llibreria utilitza les resistències internes de *pull-up* i addicionalment, garanteix que tots els pins de les columnes no utilitzats tinguin una alta impedància.

Funcions utilitzades:

1. Creació de teclat personalitzat.

```
Keypad(makekeymap(userkeymap), row[], col[], rows, cols)
```

Funció 3.1. Creació d'un teclat personalitzat.

Aquesta funció ens permet crear un teclat virtual que simularà el nostre teclat físic. Per tal d'utilitzar-la, haurem de primer donar-li un nom al nostre teclat virtual (en aquest cas serà *userKeymap*) Seguidament haurem de declarar 5 variables:

- **userKeymap**: Variable de tipus *char*. Es tracta d'una matriu de dimensions FILESxCOLUMNES on els valors de FILES i COLUMNES dependran de les dimensions del nostre teclat (en aquest cas 4x4).

```
char userkeymap[rows][cols] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
```

Funció 3.2. Exemple de declaració d'una variable per determinar un teclat virtual 4x4 (0-9, A-D, *, #).

- **row[ ]:** Variable de tipus *byte.* Es tracta d'un vector de dimensió FILES on es declararen els pins digitals en els quals es trobaran les files (d'esquerra a dreta, els 4 primers cables del teclat).
- **col[ ]:** Variable de tipus *byte.* Es tracta d'un vector de dimensió COLUMNES on es declararen els pins digitals en els quals es trobaran les columnes (d'esquerra a dreta, els 4 últims cables del teclat).

```
byte row[rows] = {0,1,2,3};
byte col[cols] = {4,5,6,7};
```

Funció 3.3. Exemple de declaració de variables per a la distribució dels pins de les files i les columnes.

- **rows:** Constant de tipus *byte*. Defineix el nombre de files que volem que tingui el teclat (en el nostre cas, 4).
- **cols:** Constant de tipus *byte*. Defineix el nombre de columnes que volem que tingui el teclat (en el nostre cas, 4).

2. Obtenció de la tecla pressionada.

```
get key ()
```

Funció 3.4. Obtenció de la tecla pressionada.

La implementació d'aquesta funció ens permet obtenir el valor de l'ultima tecla que hem pressionat i poder realitzar el menú d'opcions amb bucles condicionals al pressionar un dels polsadors del teclat.

3.3.4.2. Pantalla de cristall líquid adaptada per a mòdul I2C (LiquidCrystal_I2C.h)

Descripció:

Es tracta d'una modificació de la llibreria estàndard d'Arduino *LiquidCrystal.h per adaptar-la a* la utilització de mòduls I2C que s'utilitzen per simplificar el cablejat de connexions LCD-Arduino. Aquesta llibreria creada per Frank de Brabander i revisada per Marco Schwartz ens permet controlar *displays* I2C amb funcions pràcticament iguals a les que presenta la llibreria d'LCD d'Arduino.

Funcions utilitzades:

1. Declaració de la funció que permet manipular l'LCD utilitzat.

```
liquidCrystal_I2C ()
```

Funció 3.5. Declaració de l'LCD.

Aquesta funció ens permet ajustar les característiques que definiran el nostre LCD mitjançant els següents paràmetres:

- La posició de memòria en la que es troba guardat el nostre LCD (en aquest cas 0x27 h).

- La resolució del nostre LCD (en aquest cas 16x2).

```
liquidCrystal_I2C lcd (0x27, 16, 2);
```

Funció 3.6. Exemple de declaració d'un LCD de dues files amb 16 posicions cadascuna.

2. Neteja del display.

```
lcd.clear ()
```

Funció 3.7. Neteja del display.

Aquesta funció ens permet esborrar el contingut que hi havia escrit prèviament al nostre LCD. Es recomana utilitzar-la sempre abans d'escriure qualsevol nou text.

3. Adreçament del punter.

```
lcd.setCursor ()
```

Funció 3.8. Adreçament del punter.

Funció que permet escriure en la posició desitjada del display. Les entrades d'aquesta funció són la posició de la columna i fila d'on volem localitzar el punter per a que imprimeixi a partir d'aquest punt. Cal tenir en compte que el primer dels valors indicarà la posició de les columnes (0 fins a 15) i el segon la posició de les files (0 o 1) en el cas d'un display 16x2.

```
lcd.setCursor (5,1)
```

Funció 3.9. Exemple d'un adreçament del punter de l'LCD a la posició 5 de la segona fila.

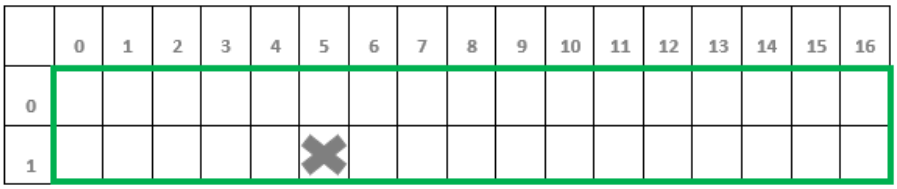


Figura 3.11. Representació gràfica de la posició 5,1 (creu gris) a l'LCD (en verd)

4. Escriure sobre el display.

```
lcd.print()
```

Funció 3.10. Escriptura sobre el *display*.

Aquesta funció simplement imprimeix sobre el display una variable que haguem declarat prèviament o un text, sempre que aquest vagi iniciat i finalitzat per cometes. Cal tenir en compte les limitacions del nostre display, ja que no podrem imprimir en la mateixa pantalla mes de 32 caràcters (16 per fila).

5. Inicialització del display.

```
lcd.init()
```

Funció 3.11. Inicialització del *display*.

Funció que invoquem en el procés de *setup* per inicialitzar el display de manera que el puguem fer servir per mostrar informació.

6. Llums de fons.

```
lcd.backlight()
```

Funció 3.12. Aplicar llums de fons al *display*.

Ja que els LCDs no generen llum per ells mateixos, aquesta funció permet aplicar un llum de fons que actua de contrast amb la llum del text variant la seva intensitat mitjançant un potenciòmetre.

3.3.4.3. CAN bus (mcp_can.h)

<u>Descripció:</u>

Aquesta llibreria es la que ens permetrà controlar el protocol de comunicacions CAN. Creada i revisada per la empresa Seeed Studio, aquesta llibreria presenta tot tipus de funcions per manejar el protocol d'una manera comprensible.

**NOTA: Tota llibreria d'Arduino presenta un arxiu .h on es recopilen totes les definicions de la llibreria, bàsicament es un llistat amb tot el que hi ha disponible a aquella llibreria. Ens molts dels casos, els autors de les llibreries es reserven el dret de nombrar algunes variables com a privades per tal de*

facilitar el treball a l'hora de cridar certes funcions. D'igual manera passa amb les funcions, en trobem de caràcter públic i de caràcter privat, aquestes últimes solen ser funcions primàries que s'utilitzen a l'hora de cridar alguna de les funcions públiques.

En aquest cas s'ha utilitzat, a part de les públiques, una de les funcions de caràcter privat, ja que per l'ús d'aquesta llibreria els autors no han cregut convenient declarar-la com a pública. Tot i així, i com era de l'interès del treball, s'ha aconseguit utilitzar. Malgrat inicialment es dubtà del correcte funcionament, el procediment va ser simple: es va esborrar la funció d'interès de l'apartat de funcions privades i es va incloure a l'apartat d'ús públic de la llibreria. Es va validar el correcte funcionament del conjunt de la llibreria i es va continuar treballant amb aquesta nova llibreria modificada, permetent-ne així l'ús d'una funció molt interessant a l'hora de treballar amb errors com més endavant veurem.

Funcions utilitzades:

1. Declaració del pin *Chip Select.*

```
MCP_CAN CAN()
```

Funció 3.13. Declaració el pin CS del CAN.

Aquesta funció declara en quin pin hem connectat a l'Arduino la sortida del pin CS del CAN. De manera que es pugui establir una correcta comunicació.

2. Inicialització del protocol CAN.

```
CAN.begin()
```

Funció 3.14. Inicialització del CAN.

Aquesta funció permet inicialitzar el protocol amb un determinat *baud rate* (nombre d'unitats de senyal per segon, en aquest cas bits per segon). Els valors d'aquest *baud rate* ja estan predefinits per l'autor i es troben en l'arxiu mcp_can_dfs.h. La llibreria ens permet treballar a 5, 10, 20, 25, 31.25, 33, 40, 50, 80, 83.3, 95, 100, 125, 200, 250, 500, 666 o 1000 KBits/s. En aquest cas s'han utilitzat 500 Kbits/s.

```
CAN.begin(CAN_500KBPS)
```

Funció 3.15. Exemple d'utilització de la funció d'inicialització del protocol amb *baud rate* = 500 Kbits/s.

3. Comprovació de si s'ha rebut missatge.

```
MCP_CAN::checkReceive()
```

Funció 3.16. Comprovar si s'ha rebut algun missatge.

El que permet aquesta funció es veure, en tot instant, si s'ha rebut algun missatge (trama). Normalment va acompanyat d'una condició que, combinant d'altres variables, fan d'aquesta una funció molt interessant. Es el cas del següent exemple:

```
if (CAN_MSGAVAIL == CAN.checkReceive())
```

Funció 3.17. Exemple de la funció de comprovació de missatge rebut. En aquest cas la condició *if* només es compleix quan hi hagi un missatge disponible al bus i hagi sigut rebut pel CAN.

4. Llegir el *buffer d'informació del bus.*

```
MCP_CAN::readMsgBuf(byte *len, byte buf[])
```

Funció 3.18. Lectura del buffer de la trama CAN.

Al implementar aquesta funció serem capaços de llegir la trama que ha sigut enviada pel bus de comunicacions. Aquesta funció requereix de dues variables de tipus byte:

- ***len:** que dependrà de la longitud del nostre missatge.
- **buf [ ]:** *Buffer* en format de vector on guardarem el camp de dades del nostre missatge.

5. Obtenció del identificador (ID) d'un missatge rebut.

```
MCP_CAN::getCanId()
```

Funció 3.19. Obtenir el identificador d'un missatge rebut.

Aquesta funció retorna l'identificador (ID) del node del qual prové el missatge.

6. Enviar el *buffer* d'informació del bus

```
MCP_CAN::sendMsgBuf(unsigned long id, byte ext, byte len,
                    const byte *buf, bool wait_sent)
```

Funció 3.20. Enviar el buffer de la trama CAN.

Aquesta funció permet construir i enviar una trama CAN pel bus de comunicacions. Per tal de poder-la implementar, es requereix declarar 4 variables:

- **id**: Variable de tipus *unsigned long* on declararem la posició de memòria en la que volem definir el nostre identificador de manera que quan realitzem una lectura de trama en un node receptor, podem identificar de quin node emissor prové aquella trama.
- **ext**: Bit d'extensió de l'identificador. Dominant (0) es definirà com a format base i forçant-lo a 1 es definirà com a estes. Diferencia trames de les dos versions de protocol estandarditzats.
- **len:** Codi de longitud de dades. Nombre de bytes de dades en el missatge. Podem definir-lo entre 0 i 8 bytes com a màxim.
- ***buf:** Camp de dades. En aquesta variable definirem el camp de dades del missatge a transmetre que es guardarà en un *buffer*.

```
unsigned char stmp [8] = {ledHIGH, 1, 2, 3, ledLOW, 5, 6, 7};
                    CAN.sendMsgBuf(0x70, 0, 8, stmp);
```

Funció 3.21. Exemple d'emissió d'una trama CAN amb ID: 0x70, Standard IDE, longitud de dades de 8 bytes i missatge a enviar "stmp": encendre un cert led, 1,2,3, apagar led, 5,6,7.

Podem afegir addicionalment altra informació interesant a la trama dins del mateix *void* com per exemple:

- **wait_send:** una variable de tipus booleana que sigui transportada pel bus de comunicacions i que contingui la informació, per exemple, d'esperar a que es retorni el missatge enviat quan aquesta es trobi en nivell alt.
- **rtrBit:** Un bit de petició de transmissió remota. Dominant (0) per a trames de dades i recessiu (1) per a trames de peticions remotes.

7. Llegir el valor d'un registre del controlador.

```
mcp2515_readRegister(const byte address)
```

Funció 3.22. Lectura d'un registre del controlador.

Aquesta funció, per defecte d'ús privat (llegir *NOTA al principi del capítol 3.3.4.3) ens permet accedir a la informació que hi ha emmagatzemada a qualsevol dels registres utilitzats pel controlador. Únicament hem de definir el valor de l'adreça del registre el qual volem llegir. Per un tema d'ordre i facilitat, els autors d'aquesta llibreria s'han reservat el dret de declarar i nombrar les posicions de memòria en la que es troben els registres amb les quals treballa el controlador. Aquestes posicions es poden trobar en l'arxiu mcp_can_dfs.h.

```
byte CErrorEFLG = CAN.mcp2515_readRegister(MCP_EFLG)
```

Funció 3.23. Exemple de la lectura del registre *Error Flag* del controlador que correspon a la posició de memòria 0x2D.

3.3.4.4. SPI (SPI.h)

Aquesta llibreria permet comunicar-se entre dispositius utilitzant la interfície SPI, essent l'Arduino el dispositiu mestre.

SPI (Serial Peripherial Interface) és un protocol de dades en sèrie síncron utilitzat pels microcontroladors per comunicar-se ràpidament amb un o més dispositius perifèrics a distàncies curtes. També es pot utilitzar per a la comunicació entre dos microcontroladors.

Amb una connexió SPI sempre hi ha un dispositiu mestre (normalment un microcontrolador) que controla els dispositius perifèrics esclaus.

Aquesta llibreria es utilitzada per l'anteriorment vista mcp_can.h per tal d'establir una connexió entre l'Arduino i el controlador CAN.

3.3.4.5. Cablejat (Wire.h)

Aquesta llibreria es la que ens permet comunicar-nos amb dispositius I2C/TWI. Per tant, per poder comunicar l'Arduino amb el *display* que incorpora un dispositiu I2C haurem d'incloure aquesta llibreria al nostre *sketch.* Utilitzant així els pins SDA (*data line* o línia de dades) i SCL (*clock line* o línia de rellotge).

3.4. Entorn i proves de laboratori

En aquest apartat es pretén explicar al lector la posada en marxa de l'equip prèviament descrit. Per a tal, aquest apartat es dividirà en quatre seccions: una primera on es descriurà **quin** es l'objectiu al qual es pretén arribar, una segona secció on s'explicarà **com** s'ha abordat aquest objectiu, una tercera on es presentaran possibles **alternatives** per assolir l'objectiu i finalment una quarta secció on es resumiran els **resultats i conclusions** extretes del laboratori.

3.4.1. Objectiu de les proves de l'equip

L'objectiu principal del projecte és construir un equip de caràcter didàctic, manejable i portable capaç de realitzar una correcta comunicació de dades mitjançant el protocol CAN, prèviament descrit, en un entorn de treball amb alt contingut d'emissions electromagnètiques.

En primer lloc, es realitza un prototip de l'equip i es confirma el seu correcte funcionament en un entorn de treball sense interferències i amb un bus constituït per dos cables *dupont*.

Un cop realitzada questa comprovació, es procedeix a la posada en marxa del sistema en un entorn amb altes emissions electromagnètiques que generen interferències sobre el bus. En aquest cas s'ha optat per realitzar el testeig al laboratori d'electrònica de l'escola on s'encendran els motors disponibles per generar interferències electromagnètiques (EMI).

Assolit aquest objectiu i demostrada la robustesa del protocol, es procedeix a sotmetre l'equip a situacions d'estrès que dificultin, destorbin o impedeixin al bus la correcta comunicació, per tal d'analitzar com de fiable arriba a ser el protocol i quins mecanismes de gestió d'errors implementa.

3.4.2. Desenvolupament de l'objectiu

Per tal de generar situacions hostils per al protocol, s'han tingut en compte dues variables que afecten sobre la fiabilitat del protocol, bé sigui atacant directament el bus o indirectament dificultant les comunicacions tal i com s'explica a continuació:

- Actuar sobre els **motors**: Al laboratori d'electrònica de l'escola es disposa de dos motors elèctrics, un de corrent continua i un altre d'alterna, com es lògic, tenint mes o menys motors encesos, tindrem mes o menys soroll electromagnètic.
  A més, al laboratori també es disposa d'un conjunt de bombetes de llum en sèrie amb els seus corresponents interruptors d'engegada i encesa. Per tan s'han classificat i recollit en la Figura 3.12 els experiments segons el grau o nivell d'hostilitat que provoquen sobre l'equip.

GRAU D'HOSTILITAT 1	GRAU D'HOSTILITAT 2	GRAU D'HOSTILITAT 3
S'introdueix un únic motor en marxa (Continu o Altern)	S'introdueixen tots dos motors en marxa a l'hora.	A més dels dos motors en marxa, es fa passar el bus entre les bombetes, engegant-les i apagant-les

Figura 3.12. Experiments realitzats amb els motors segons el grau d'hostilitat que generen.

Figura 3.13. Assaig de laboratori testejant l'equip en els graus d'hostilitat mes alts. A la imatges es pot apreciar com els dos cables de coure disposats en paral·lel i formant debanats, es fan passar entre d'altres debanats del cable connectat a les bombetes. Sota la taula es troben els dos motors.

- Actuar sobre els **cables**: Els cables son un altre factor clau a l'hora de complicar la feina al protocol, ja que podem actuar directament sobre ells fent-los mes vulnerables, ja sigui augmentant la seva distancia, utilitzant cables poc protegits o disposar-los formant debanats entre si i entre d'altres elements. La classificació del grau d'hostilitat actuant sobre els cables es recull a la Figura 3.14:

GRAU D'HOSTILITAT 0	GRAU D'HOSTILITAT 1	GRAU D'HOSTILITAT 2	GRAU D'HOSTILITAT 3
2 cables de tipus *upon* o *jumper*.	2 cables de telefonia, parell trenat i amb cobertura de protecció.	2 cables de coure, parell trenat.	2 cables de coure, en paral·lel.
Longitud dels cables: 10 cm.	Longitud dels cables: 2 m.	Longitud dels cables: 5 m.	Longitud dels cables: 7 m.
Disposició: la seva poca longitud no permet cap disposició.	Disposició: Al costat dels motors.	Disposició: Rodejant els motors.	Disposició: realitzant varis debanats sobre ells mateixos i fent passar entre aquests, altres debanats del cable connectat a les bombetes.

Figura 3.14. Experiments realitzats amb els cables segons el grau d'hostilitat que generen.

Figura 3.15. Diferents tipus de cables emprats durant les proves de laboratori

3.4.3. Alternatives per desafiar la robustesa de l'equip

Addicionalment, s'han plantejat altres opcions externes al medi per tal de "forçar" errors en aquest sistema de comunicacions. Algunes d'aquestes alternatives podrien formar part d'una possible ampliació d'aquest treball i són les següents:

- **Desconnexió física d'un dels terminals del bus.** Desconnectant un dels dos cables que formen el bus (CAN H o CAN L) per uns segons forçarà que el controlador emeti errors de transmissió. Aquesta primera proposta es realitza per confirmar que efectivament l'equip està preparat per detectar errors i avisar-ne i que el fet de que no en mostri no es deu a cap altra raó que la immunitat a les radiacions electromagnètiques que presenta el protocol.
- **Inclusió d'un tercer microcontrolador.** Afegint una tercera placa microcontroladora (per exemple una Raspberry Pi o un PIC (*Programable Integrated Circuit*) s'augmentaria el nombre de nodes al bus i per tan dificultaria el procés de transmissió, addicionalment aquest microcontrolador podria intervenir sobre el bus durant un període de temps determinat generant certes trames d'error.
- **"Placa generadora d'errors".** Es tracta de dissenyar i construir una placa per a provocar l'aparició d'errors esporàdics al bus, per exemple curtcircuitant-lo. Es proposa com a possibilitat, construir un placa basada en un dispositiu del tipus LM555 actuant com a oscil·lador astable sobre el bus. D'aquesta manera i realitzant els càlculs corresponents de resistències i condensadors, s'obtindria una determinada freqüència que per a un cicle de treball desitjat, curtcircuitaria el bus durant un temps determinat, generant així errors de transmissió.

3.4.4. Resultats de les proves

Un cop realitzades totes les proves d'hostilitat descrites a la Figura 3.12 i la Figura 3.14**,** es pot confirmar que el bus es molt robust vers les interferències electromagnètiques i es pot afirmar que l'equip es pràcticament invulnerable a la radiació electromagnètica que generen fins a dos motors elèctrics en funcionament ja que no ha presentat errors durant cap de les proves d'estrès a les que ha sigut somes.

De fet, un estudi realitzat a la universitat de Missouri (*Fei Ren; Y.R. Zheng; Maciej Zawodniok; J. Sarangapani* [6]) on es va estudiar com afecten les interferències electromagnètiques sobre el rendiment del bus CAN, confirma les nostres conclusions. En els experiments, els autors van utilitzar diferents tipus de cables, estructures i CAN *bit rates* per tal d'analitzar aquests efectes. Les proves mostren que els polsos d'EMI, que es troben freqüentment a l'industria automobilística i en d'altres maquinaries, pot provocar la reducció dels *bit rates* i errors en comunicacions CAN d'alta velocitat.

L'estudi assegura que la major causa de degradació del rendiment de la comunicació CAN (en relació a les EMI) ve donada per les carregues inductives. Per tal de demostrar-ho, els autors opten per recrear EMI mitjançant un relé magnètic (gran carrega inductiva) junt a un interruptor que obrirà o tancarà el pas a les EMI les quals seran després acoblades amb la comunicació del bus. Aquests polsos acoblats causen una reducció en la velocitat de transmissió i errors de comunicació.

Les proves van constar de varis experiments variant: el tipus de cable (recoberts i no recoberts) i l'estat del interruptor ON-OFF (40 vegades/s i 60 vegades/s). Addicionalment van variar les longituds dels cables.

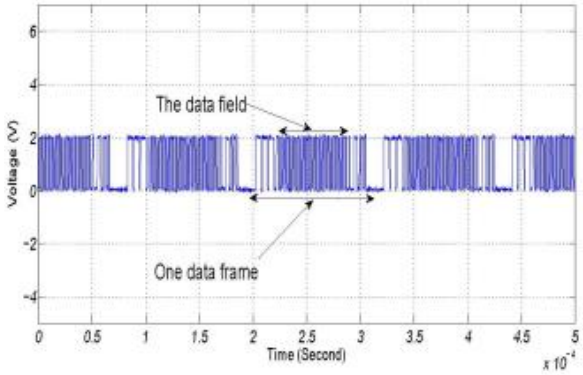


Figura 3.16. Trames CAN de missatges amb *bit rate*: 1Mbps (Font: Fei Ren; Y.R. Zheng; Maciej Zawodniok; J. Sarangapani [6])

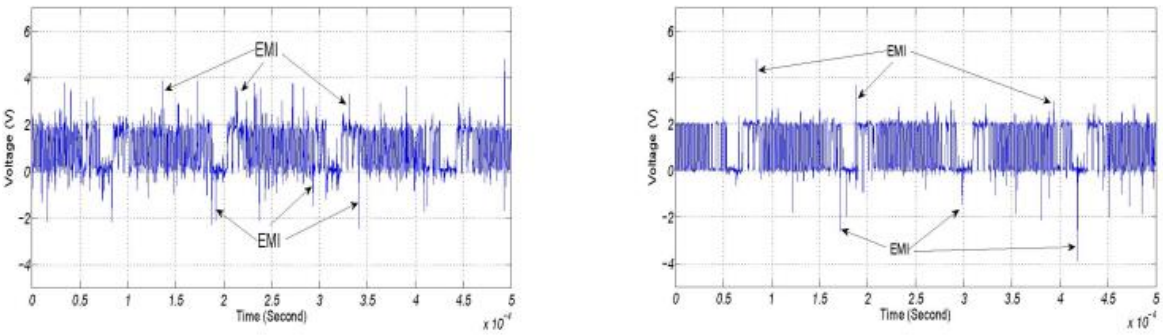


Figura 3.17. El gràfic de l'esquerra mostra les trames CAN de missatges en cables sense cobertura; a la dreta les trames CAN de missatges en cables amb cobertura (Font: Fei Ren; Y.R. Zheng; Maciej Zawodniok; J. Sarangapani [6])

És interessant analitzar que, les conclusions que van extreure els autors de l'estudi, validen i recolzen els resultats obtinguts en les proves realitzades al laboratori de l'escola. En l'article fan el següent recull de conclusions:

- Les trames de dades en cables de longitud petita tenen millor forma que les de cables llargs. Això es degut a que els cables llargs tenen mes atenuació que els curts.
- Més EMI causa més reducció del *bit rate*. Quan s'imposa un *bit rate* de 503Kbps en la situació d'un cable de 2 m i l'interruptor canvia d'estat a l'ordre de 2 vegades/s, la reducció del *bit rate* actual es de 0.096 Kbps, es a dir 0.02% reduït del *bit rate* actual. Quan l'interruptor canvia d'estat a ordre de 3 vegades/s la reducció es de 0.146 Kbps es a dir un 0.03%.
- La utilització de cables recoberts pot protegir, en gran proporció, la comunicació CAN de les interferències electromagnètiques.
- El nombre de nodes que confina el bus pot arribar a afectar, a petita escala, el *bit rate*.

4. Anàlisi de l'impacte ambiental

El treball que s'ha realitzat, pel seu propi abast i dimensions que comporta, no suposa un impacte ambiental que s'hagi de tenir necessàriament en compte. Sí es important, però, recollir un breu anàlisi del possible impacte ambiental tant en el seu desenvolupament, com en funcionament normal i sota avaria i finalment en el seu procés de reciclatge.

L'equip resultant del treball està construït amb elements de petites dimensions i baix consum d'energia elèctrica que pot ser subministrada de diferents formes, la opció més correcta des d'un punt de vista de consciencia ambiental seria mitjançant un únic ordinador que implementés dos ports USB tipus 2.0, ja que l'ús de piles o bateries que contenen substancies tòxiques com el mercuri, liti, plom o níquel poden ser altament perjudicials (a part que difícilment reciclables) per a la salut dels essers vius i l'atmosfera. Els components electrònics s'han adquirit majoritàriament mitjançant compra on-line en una única comanda a excepció del material per elaborar la caixa protectora. Els embolcalls de plàstic i cartró dels elements s'han reciclat adequadament, seguint la normativa local establerta.

L'equip ha estat pensat per un funcionament en períodes de temps reduïts, inicialment per realitzar les proves previstes al treball i per la seva utilització en accions didàctiques de laboratori. Durant el seu funcionament normal, l'equip produeix radiacions electromagnètiques de molt baixa intensitat que no suposen cap risc per al medi ni les persones o animals. Addicionalment, s'ha tingut en compte en el disseny de l'equip que aquest ha de poder servir per a possibles futures ampliacions de l'estudi del protocol CAN o a d'altres estudiants, deixant espai per incloure un node més a la xarxa, per tal "d'amortitzar" en mesura del possible l'eventual impacte ambiental.

En situació d'avaria o mal funcionament de l'equip caldrà revisar el circuit de connectivitat i funcionament dels diferents components per tal de procedir a la seva reparació o substitució.

En el procés de reciclatge de l'equip, caldrà atendre la normativa vigent per tal de que cadascun dels elements que el composen tinguin el mínim impacte ambiental possible.

5. Conclusions

El treball s'ha dut a terme segons la planificació temporal prevista, recollint els requeriments establerts i assolint els objectius plantejats.

També ha suposat un repte professional i personal gratificant ja que m'ha permès compaginar la gestió integra d'un projecte d'enginyeria amb la seva execució; des de la concepció inicial fins al seu lliurament, passant per totes les fases del seu desenvolupament.

Un cop finalitzat aquest projecte podem afirmar amb alta certesa, per una banda, que s'han aconseguit els objectius inicials de tipus didàctic, és a dir, l'enriquiment en coneixements de l'àmbit dels sistemes de comunicacions, concretament sobre xarxes CAN, així com el notori progrés en programació en llenguatge d'alt nivell C++.

Per altra banda, s'ha aconseguit construir un equip electrònic que implementa una xarxa CAN de dos nodes per realitzar transmissions de dades de forma controlada mitjançant el software desenvolupat específicament per als microcontroladors programables inclosos. Addicionalment s'ha aconseguit un objectiu molt interesant que es que aquest equip ofereixi un grau de flexibilitat de treball i mobilitat molt adient per a possibles finalitats educatives de laboratori, com pot ser per exemple la inclusió en una pràctica de laboratori d'una assignatura que tractés en certa mesura les comunicacions, per exemple en graus d'electrònica o telecomunicacions, entre d'altres. Així mateix també permet a futurs estudiants una possible ampliació de les seves característiques.

Finalment i en referencia al testeig de l'equip en entorns industrials amb alt contingut d'emissions electromagnètiques, s'han obtingut els resultats esperats en la seva confecció i control, verificant l'alta robustesa que presenta aquest tipus de xarxes de comunicacions vers les interferències electromagnètiques, tal i com es desenvolupa en l'apartat de resultats de les proves (3.4.4).

6. Pressupost

El pressupost que es presenta a continuació conté la totalitat del projecte des de la perspectiva econòmica. Això inclou: components utilitzats, hores de disseny, hores de programació, hores de muntatge, hores de laboratori, hores de redactat i desplaçament entre d'altres.

6.1. Costs de materials

En la següent taula es recullen tots els materials emprats en la confecció de l'equip:

CONCEPTE	QUANTITAT	PREU UNITARI	PREU TOTAL
LCD	2	5,24	10,48
CAN MODULE	4	3,5	14
LCD-I2C	2	2,5	5
TECLAT 4X4	2	5,99	11,98
PROTOBOARD	1	5,99	5,99
PACK DE CABLES	2	6,99	13,98
CONNECTORS BERG	3	0,5	1,5
PLANXA METACRILAT DIN A4	2	4,6	9,2
PLANXA METACRILAT DIN A3	3	8,25	24,75
CARGOLS I FEMELLES			2,62
ABRAÇADERES			3,25
FRONTISA			4,5
ARDUINO MEGA	2	13,99	27,98
SUBTOTAL MATERIAL			135,23

El pressupost parcial del material utilitzat en la confecció de l'equip es de CENT TRENTA-CINC EUROS I VINT-I-TRES CÈNTIMS.

6.2. Costs de mà d'obra

Les activitats de muntatge i testeig de l'equip estan previstes de ser realitzades per un tècnic especialista amb un sou base de 15 €/h.

RECURS HUMÀ	HORES	SOU (€/h)	COST (€)
TÈCNIC ESPECIALISTA	40	15	600

El pressupost parcial de muntatge i test de l'equip es de SIS-CENTS EUROS.

6.3. Costs d'enginyeria

El temps de realització d'aquest projecte ha sigut de 20 setmanes a temps setmanals de 30 hores i amb un sou base estimat per a un enginyer tècnic de 18 €/h. Aquestes hores inclouen entre d'altres el disseny electrònic i mecànic, la programació i el redactat.

RECURS HUMÀ	HORES	SOU (€/h)	COST (€)
ENGINYER TÈCNIC	600	18	10800

El pressupost parcial de l'enginyeria de disseny del projecte elaborat es de DEU MIL VUIT-CENTS EUROS.

6.4. Cost total

El pressupost total del projecte es calcula de la següent forma:

PRESSUPOST DEL MATERIAL	135,25 €
PRESSUPOST DEL MONTATGE I TEST	600 €
PRESSUPOST D'ENGINYERIA	10.800 €
COSTS D'ACTIVITATS VARIES	50 €
PREU TOTAL SENSE IMPOST	11.585,25 €
PREU TOTAL AMB IMPOST (IVA = 21%)	14.018,15 €

Per costs d'activitats varies es refereix al conjunt d'activitats com poden ser el tall de planxes de metacrilat mitjançant làser, les impressions del document i/o els desplaçaments realitzats.

El pressupost total del projecte ascendeix a la quantitat de CATORZE-MIL DIVUIT EUROS I QUINZE CÈNTIMS.

Bibliografia

[1] Arduino MEGA 2560 REV3. A: *Arduino* [en línia]. [Consulta: 6 novembre 2019]. Disponible a: <https://store.arduino.cc/arduino-mega-2560-rev3>.

[2] CAN-BUS Shield V2.0. A: *Seed Technology Inc* [en línia]. [Consulta: 27 setembre 2019]. Disponible a: <http://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/>.

[3] Curià, R. [Mephirius] (21 novembre, 2019). *Problem Connecting 2 arduinos with CAN BUS.* [Publicació fòrum en línia]. Disponible a: <https://forum.arduino.cc/index.php?topic=649446.msg4379695#msg4379695>.

[4] Curià, R. [Mephirius] (2 desembre, 2019). How to extract a *CAN bus frame.* [Publicació fòrum en línia]. Disponible a: <https://forum.arduino.cc/index.php?topic=651360.0>.

[5] Curià, R. [Mephirius] (10 desembre, 2019). *How to use a private function from .h archive*. [Publicació fòrum en línia]. Disponible a: <https://forum.arduino.cc/index.php?topic=653071.msg4401559#msg4401559>.

[6] Fei Ren; Y.R. Zheng; Maciej Zawodniok; J. Sarangapani. *Effects of electromagnetic interference on control area network performance* [en línia]. Rolla: University of Missouri-Rolla. Departament of Electrical & Computer Engineering, 2007. Disponible a: <file:///C:/Users/Roger/Downloads/EffectsOfElectromagneticInterference_09007dcc8054c036%20(2).pdf>.

[7] History of CAN technology. A: *CAN-CIA* [en línia]. [Consulta: 10 octubre 2019]. Disponible a: <https://www.can-cia.org/can-knowledge/can/can-history/>.

[8] International Organization for Standarization. *ISO 11898-1: Road vehicles – Controller area network (CAN). Part 1: Data link layer and physical signaling* [en linia]. Geneva, 2003 [Consulta 9 setembre 2019]. Disponible a: <http://read.pudn.com/downloads209/ebook/986064/ISO%2011898/ISO%2011898-1.pdf>.

[9] International Organization for Standarization. *ISO 11898-2: Road vehicles – Controller area network (CAN). Part 2: High-speed medium acces unit* [en línia]. Geneva, 2003 [Consulta 9 setembre 2019]. Disponible a: <http://read.pudn.com/downloads557/ebook/2297674/ISO11898/ISO11898-2.pdf>.

[10] Llamas, L. "Usar un teclao matricial con arduino" [apunt de blog]. A: Luisllamas [en linia]. 2 octubre 2016 [Consulta: 1 octubre 2019; 16:22 CET]. Disponible a: <https://www.luisllamas.es/arduino-teclado-matricial/>.

[11] López Fresno, J. Nodo de comunicación basado en el bus CAN [en línia]. Treball final de grau, URV, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona. Departament d'Enginyeria Electrònica, Elèctrica i Automàtica, 2004 [Consulta 25 Setembre]. Disponible a: <http://deeea.urv.cat/public/PROPOSTES/pub/pdf/561pub.pdf>.

[12] Microchip Technology Inc. *Introduction to Controller Area Network (CAN)* [Diapositives]. Disponible a: <https://www.microchip.com/stellent/groups/SiteComm_sg/documents/DeviceDoc/en558265.pdf>.

[13] Modulo Adaptador LCD a I2C. A: *Naylampmechatronics* [en linia]. [Consulta: 1 octubre 2019]. Disponible a: <https://naylampmechatronics.com/displays/60-modulo-adaptador-lcd-a-i2c.html>.

[14] Placa de pruebas. A: *Wikipedia* [en linia]. Wikimedia Fundation, 2019. [Consulta: 1 octubre 2019]. Disponible a: <https://es.wikipedia.org/wiki/Placa_de_pruebas>.

[15] Referències bibliogràfiques. A: *UPC* [en línia]. Servei de Biblioteques, Publicacions i Arxius, 2019. [Consulta: 20 desembre 2019]. Disponible a: <https://bibliotecnica.upc.edu/investigadors/referencies-bibliografiques>.

[16] Richards, P. *A CAN Physical Layer Discussion* [en línia]. USA: Microchip Technology Inc., 2002. DS00228A [Consulta 10 octubre 2019]. Disponible a: <http://ww1.microchip.com/downloads/en/appnotes/00228a.pdf>.

[17] Vazquez M. *Introdución a CAN BUS* [Diapositives]. 2016 [Consulta 5 octubre 2019] Disponible a: <https://www.slideshare.net/MartnVzquez2/introducin-a-can-bus>.

Annex A : Codi del programa

A1. Codi del microcontrolador mestre

```
          // Incluir llistat de llibreries utilitzades:
#include <LiquidCrystal_I2C.h>                          // LCD + I2C
#include <Keypad.h>                                     // Teclat
#include <mcp_can.h>                                    // CAN BUS
#include <SPI.h>                                        // Comunicació SPI
#include <Wire.h>                                       // Comunicació I2C

/*SAMD core*/                                          // Definició de MCU

#ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
  #define SERIAL SerialUSB
#else
  #define SERIAL Serial
#endif

const int SPI_CS_PIN = 10;

MCP_CAN CAN(SPI_CS_PIN);                          // Fixa el pin CS a 10.
int conta;
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
    { '1','2','3','A'},
    { '4','5','6','B'},
    { '7','8','9','C'},
    { '*','0','#','D'}
};
byte rowPins[ROWS] = {0, 8, 2, 3};
byte colPins[COLS] = {4, 5, 6, 7};
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins,
ROWS, COLS);    // Declaració de teclat matricial 4x4 personalitzat.
LiquidCrystal_I2C lcd (0x27, 16, 2);               // Declaració del LCD.
//=================================================================
void menu_pantalla0()                           // Pantalla 0: benvinguda.
{
  lcd.setCursor(0, 0);
  lcd.print("Welcome to ROGER");
  lcd.setCursor(0, 1);
  lcd.print("CURIA TFG");
  delay(2000);
}

//=================================================================
void menu_pantalla1()                        // Pantalla 1: menú principal.
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1)START 2)STOP");
  lcd.setCursor(0, 1);
```

```
  lcd.print("3)ERROR 4)STATUS");
}
//=================================================================
void setup()
{
    SERIAL.begin(115200);          // Inicialització del port serial.
Fixa la velocitat de mostrieg de dades a 115200 bps.

    while (CAN_OK != CAN.begin(CAN_500KBPS))     // Inicialització del
CAN BUS. Baudrate = 500kbps,
    {
        SERIAL.println("CAN BUS Shield init fail");
        SERIAL.println(" Init CAN BUS Shield again");
        delay(100);
    }
    SERIAL.println("CAN BUS Shield init ok!");
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  conta = 0;
  menu_pantalla0();
  menu_pantalla1();
}
unsigned long msg_count = 0;            // Contador de missatges enviats.
unsigned int contError = 0;             // Contador de detecció d'error
Tx != Rx.
unsigned char stmp[] = {0};             // Buffer d'informació enviada a
la trama del bus.
unsigned char temp = 0;                 // variable temporal per guardar
la informació enviada pel buffer i poder sincronitzar amb la rebuda.
bool START = 0;
void loop()
{
  char Key = customKeypad.getKey();
  if (Key =='1'){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("TRANSMITTING...");
    lcd.setCursor(0,1);
    lcd.print("2) PAUSE");
        while (Key != '2' && msg_count <= 13199){
         unsigned char len2 = 0;
         unsigned char buf2[1];
           if (CAN_MSGAVAIL == CAN.checkReceive())       // Comprovar si
hi ha algun missatge disponible al CAN BUS.
            {
              CAN.readMsgBuf(&len2, buf2);               // LLegeix la
trama rebuda.
              unsigned long CanId2 = CAN.getCanId();     // Obtenció del
ID pel qual ha sigut rebut el missatge.
              //Serial.println("=========================================
=====");     // --Per tal de visualitzar els resultats pel monitor serial
en cas de fallida d'algun
              //Serial.println("MSG SENT FROM CANID 0x70");    // LCD,
converteixi els següents comentaris en codi.--
              //Serial.println(temp);
             // presentar info missatge enviat pel CANID TX
              //Serial.println("MSG RECEIVED FROM CANID 0x");
```

```
              //Serial.println(CanId2, HEX);
              //Serial.println("MSG RECEIVED:");
              //Serial.print(buf2[0]);
              // presentar info missatge rebut del CANID RX
              //Serial.println();
              //_____________________________ERRORS
PART_____________________________
              byte CErrorTEC = CAN.mcp2515_readRegister(MCP_TEC);
            // Lectura del buffer TEC.
              byte CErrorREC = CAN.mcp2515_readRegister(MCP_REC);
            // Lectura del buffer REC.
              byte CErrorEFLG = CAN.mcp2515_readRegister(MCP_EFLG);
                   // Lectura del buffer EFLG.
              //Serial.print("TEC");
              //Serial.print("\t");
              //Serial.print("REC");
              //Serial.print("\t");
              //Serial.println("EFLG");
              //Serial.print(CErrorTEC);
              //Serial.print("\t");
              //Serial.print(CErrorREC);
              //Serial.print("\t");
              //Serial.print(CErrorEFLG);
              //Serial.println();
              }
            else

              {
              CAN.sendMsgBuf(0x70,0,1,stmp);
               // Si no s'ha rebut missatge, enviar el primer missatge.
              }
              //-------------------------SEND PART-----------------------
-------------------------------------------------------------------------
--------
              CAN.sendMsgBuf(0x70,0,1,stmp);
               // Enviar missatge.
              temp = stmp[0];        // guardar informació de transmissió
              stmp[0]++;
               // Incrementar en una unitat el missatge a transmetre.
              Key = customKeypad.getKey();
              //Serial.println(Key);
              msg_count++;
               // Incrementar en una unitat el contador de missatges
enviats.
      }

  }
  if (Key == '2'){                     // --Funció quan tecla [2]
pressionada. Pausa la transmissió de dades,
    lcd.clear();
               // permet reanudar-la o finalitzar-la i presentar els
errors.--
    lcd.setCursor(2,0);
    lcd.print("TRANSMISSION");
    lcd.setCursor(5,1);
    lcd.print("PAUSED");
    delay (4000);
    lcd.clear();
```

```
    lcd.setCursor(0,0);
    lcd.print("1) RESUME");
    lcd.setCursor(0,1);
    lcd.print("3) ERRORS");
    }
  if (Key == '3'){                            // --Funció quan tecla [3]
pressionada. Mostra el nombre de missatges enviats,
    lcd.clear();
                // els errors de transmissió i recepció.--
    lcd.setCursor(0,0);
    lcd.print("Number of msgs:");
    lcd.setCursor(0,1);
    lcd.print(msg_count);
    delay(3000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("TX Errors:");
    lcd.setCursor(0,1);
    lcd.print("RX Errors:");
    lcd.setCursor(11,0);
    byte CErrorTEC = CAN.mcp2515_readRegister(MCP_TEC);
    byte CErrorREC = CAN.mcp2515_readRegister(MCP_REC);
    lcd.print(CErrorTEC);
    lcd.setCursor(11,1);
    lcd.print(CErrorREC);
    delay (4000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("4) BUS STATUS");
    lcd.setCursor(0,1);
    lcd.print("*) MENU");
    }
  if (Key == '4'){                            // Funció quan tecla [4]
pressionada. Mostra l'estat en el que es troba el BUS CAN.
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Bus status:");
    byte CErrorTEC = CAN.mcp2515_readRegister(MCP_TEC);
    byte CErrorREC = CAN.mcp2515_readRegister(MCP_REC);
    byte CErrorEFLG = CAN.mcp2515_readRegister(MCP_EFLG);
    if (CErrorTEC > 127 && CErrorTEC < 255 || CErrorREC > 127){
      lcd.setCursor(0,1);
      lcd.print("Error-Passive");
      }
    else if (CErrorTEC > 255){
      lcd.setCursor(0,1);
      lcd.print("Bus-Off");
      }
    else {
      lcd.setCursor(0,1);
      lcd.print("Error-Active");
      }
    delay (4000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("*) MENU");
  }
```

```
  if (Key == '*'){                                // Funció quan tecla [*]
pressionada. Reset de parametres i torna a menú principal.
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("PARAMETERS RESET");
    lcd.setCursor(0,1);
    lcd.print("CORRECTLY");
    msg_count = 0;
    byte CErrorTEC = CAN.mcp2515_readRegister(MCP_TEC);
    byte CErrorREC = CAN.mcp2515_readRegister(MCP_REC);
    CErrorTEC = 0;
    CErrorREC = 0;
    delay(3000);
    menu_pantalla1();
  }
}
```

A2. Codi del microcontrolador esclau

```
#include <mcp_can.h>
#include <SPI.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
/*SAMD core*/
#ifdef ARDUINO_SAMD_VARIANT_COMPLIANCE
  #define SERIAL SerialUSB
#else
  #define SERIAL Serial
#endif

const int SPI_CS_PIN = 10;
LiquidCrystal_I2C lcd (0x27, 16, 2);
void menu_pantalla0()
{ lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print("Welcome");
  delay(1000);
}

//=========================================================================
void menu_pantalla1()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WAITING FOR CAN");
  lcd.setCursor(0, 1);
  lcd.print("BUS CONNECTION");
}
MCP_CAN CAN(SPI_CS_PIN);
void setup()
{
    SERIAL.begin(115200);
    while (CAN_OK != CAN.begin(CAN_500KBPS))
    {
        SERIAL.println("CAN BUS Shield init fail");
        SERIAL.println(" Init CAN BUS Shield again");
        delay(100);
    }
    SERIAL.println("CAN BUS Shield init ok!");
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0, 0);
    menu_pantalla0();
    menu_pantalla1();

}


void loop()
{
  { //--------------------------------RECEIVE PART-----------------------
    unsigned char len = 0;
    unsigned char buf[1];
```

```
    if (CAN_MSGAVAIL == CAN.checkReceive())
      { CAN.readMsgBuf(&len, buf);
        unsigned long CanId = CAN.getCanId();
        Serial.println("=============================================");
        Serial.println("MSG FROM CANID 0x");
        Serial.println(CanId, HEX);
        Serial.println("MSG RECEIVED:");
        Serial.print(buf[0]);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("TRANSMITTING...");
        Serial.println();
        //------------------------------SEND PART------------------------
        unsigned char stmp2[1] = {buf[0]};
        Serial.println("MSG SENT FROM CANID 0x80:");
        CAN.sendMsgBuf(0x80,0,1,stmp2);
        Serial.println(stmp2[0]);

      }
  }
}
```

Annex B: Datasheets dels components

[1] Placa microcontroladora Mega 2560 (Marca ELEGOO: https://www.elegoo.com). *Datasheet* disponible a: http://www.mantech.co.za/datasheets/products/A000047.pdf

[2] *Liquid Crystal Display 16x2.* *Datasheet* disponible a: https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

[3] CAN *Controller* MCP2515. *Datasheet* disponible a: https://www.mouser.com/datasheet/2/268/20001801H-708845.pdf

CAN *Transceiver* TJA1050. *Datasheet disponible a:* https://www.nxp.com/docs/en/data-sheet/TJA1050.pdf

[4] Mòdul adaptador per I2C, PCF8574; PCF8574A. *Datasheet disponible a:* https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf

[5] Teclat matricial 4x4. *Datasheet* disponible a: http://www.electronicoscaldas.com/datasheet/Teclado-membrana-matricial-4x4.pdf

Annex C: Arxius .cpp i .h de les llibreries utilitzades

En aquest annex s'inclouen els arxius formatius i que contenen les definicions de totes les funcions de les llibreries utilitzades això inclou els arxius .cpp i .h d'aquestes. Per tal de reduir la mida d'aquest apartat (aproximadament 130 pagines) s'han adjuntat els arxius virtualment de manera que fent un simple *doble-click* (funciona per versions en Word, en altres versions es pot utilitzar l'enllaç directe) et redirigirà a l'arxiu corresponent de la llibreria desitjada.

Llibreria **mcp_can** (.ZIP disponible a: <https://github.com/Seeed-Studio/CAN_BUS_Shield>).

Versió Microsoft Word	CAN_library.docx	Versió PDF	CAN_library.pdf

Llibreria **LiquidCrystal_I2C** (.ZIP disponible a: <https://github.com/johnrickman/LiquidCrystal_I2C>).

Versió Microsoft Word	LCD_library.docx	Versió PDF	LCD_library.pdf

Llibreria **Keypad** (.ZIP disponible a: <https://github.com/Chris--A/Keypad>).

Versió Microsoft Word	KEYPAD_library.docx	Versió PDF	KEYPAD_library.pdf

Llibreria **SPI** (.ZIP disponible a: <https://github.com/PaulStoffregen/SPI>).

Versió Microsoft Word	SPI_library.docx	Versió PDF	SPI_library.pdf

Llibreria **Wire** (.ZIP disponible a: <https://github.com/esp8266/Arduino/tree/master/libraries/Wire>).

Versió Microsoft Word	WIRE_library.docx	Versió PDF	WIRE_library.pdf