Universidad Autónoma de Madrid

Escuela politécnica superior

Proyecto fin de carrera

SISTEMA DE COMUNICACIÓN CON ALTAS RESTRICCIONES DE TIEMPO Y CONSUMO ENTRE UN MAESTRO Y MÚLTIPLES ESCLAVOS

Ingeniería de Telecomunicación

Karim Kaci

Junio 2016

SISTEMA DE COMUNICACIÓN CON ALTAS RESTRICCIONES DE TIEMPO Y CONSUMO ENTRE UN MAESTRO Y MÚLTIPLES ESCLAVOS

AUTOR: Karim Kaci
TUTOR: Guillermo González de Rivera Peces

Grupo HCTLab
Dpto. de Tecnología Electrónica y de las comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2016

Resumen

La ambición jamás se detiene,
ni siquiera en la cima de la grandeza.

Napoleón Bonaparte

Resumen

Este proyecto surge como una propuesta para el proyecto internacional *BigBOSS*, liderado por el *Lawrence Berkeley National Laboratory*, Estados Unidos. Se trata de un proyecto que, utilizando el espectrógrafo BigBOSS, pretende elaborar el mayor mapa del universo realizado hasta el momento.

Para tal objetivo, BigBOSS propone utilizar cinco mil fibras ópticas ubicadas en el plano focal del espectrógrafo, apuntando cada una de ellas hacia una galaxia. Esto plantea, entre otras, la problemática del posicionamiento de las fibras. Esta tarea se compone, por una parte, de un dispositivo que sea capaz de posicionar la fibra óptica en una coordenada deseada (en adelante llamado actuador) y, por otra parte, de un sistema capaz de establecer una comunicación con dicho actuador.

El presente proyecto se centra en el diseño y el desarrollo de un sistema que permite establecer dicha comunicación entre todos los actuadores que se utilizan y la unidad central que contiene los datos relevantes para el funcionamiento del proyecto BigBOSS, en este caso, las coordenadas donde los actuadores deben posicionar las fibras ópticas. El sistema desarrollado en este proyecto debe ser capaz de transmitir los datos de la unidad central hacia los actuadores en el menor tiempo posible y con el menor consumo energético.

AGRADECIMIENTOS: Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad, a través del proyecto 'Construcción y Test del Sistema Posicionador de Fibras de DESI' de referencia AYA2014-60641-C2-2-P .

Palabras Clave

BigBoss, I^2C, Multiplexores, Ethernet, BeagleBone, Posicionadores, Actuadores

Abstract

Ambition is never content,
even on the summit of greatness.

Napoleon Bonaparte

Abstract

This project comes as a proposal for the international project *BigBOSS*, led by Lawrence Berkeley National Laboratory, US. This is a project that, using the spectrograph BigBOSS, aims to develop the largest map of the universe made so far.

For this purpose, BigBOSS propose to use five thousand optical fibers, located in the focal plane of the spectrograph, each pointing to a galaxy. This raises, among others, the problem of positioning of the fibers. This task consists, on the one hand, of a device that is capable of positioning the optical fiber in a desired coordinate (hereinafter called actuator) and, on the other hand, a system capable of establishing communication with said actuator.

This project focuses on the design and development of a system that allows such communication between all the actuators that are used and the central unit containing the relevant data for the operation of BigBOSS project, in this case, the coordinates to which the actuators must position the optical fibers. The system developed in this project should be able to transmit data from the central unit to the actuators in the shortest possible time and with the less possible energy consumption.

Acknowledgements: This work was funded by the Ministry of Economy and Competitiveness, through the project 'Construcción y Test del Sistema Posicionador de Fibras de DESI' with reference AYA2014-60641-C2-2-P.

Key words

BigBoss, I^2C, Multiplexors, Ethernet, BeagleBone, Positioners, Actuators

Agradecimientos

Alles hat ein Ende,
nur die Wurst hat zwei!

Proverbio alemán

[1]

Una etapa más de la vida llega a su fin. Se terminan las prácticas, las interminables horas de clase, los días en los que tienes dos exámenes y por la madrugada una entrega... se termina la etapa universitaria.

Me gustaría salir de ella agradeciendo, en primer lugar, a mis padres Abderrahmane e Iryna por el incondicional apoyo mostrado, la infinita paciencia que han tenido conmigo y los valiosos consejos dados a lo largo de todos estos años.

No puedo obviar a mis compañeros Jose, Luís, Miriam, Tina y Xu. Gracias por todo el tiempo y los momentos que hemos pasado juntos, gracias por hacer de ésta una experiencia única. También quiero agradecer al resto de mis amigos, Alejandro, Dani, Israel, Jose, Miriam, Nasib, Ricardo, Vero - entre otros - sin vosotros, no sería quien soy. Gracias.

Por último, pero no por ello menos importante, quiero dar las gracias a mi tutor Guillermo González de Rivera Peces y al equipo HCTLab por la posibilidad de realizar este proyecto y toda la ayuda brindada a lo largo del mismo. Agradecer también a mis profesores, en especial a Susana Holgado, Eloy Anguiano y Javier Ortega, ya que sin su ayuda y paciencia no habría llegado hasta aquí.

Karim Kaci
Junio 2016

[1] Todo tiene un final, menos la salchicha, que tiene dos!

Índice general

Índice de figuras **XII**

Índice de cuadros **XVI**

1. Introducción **1**

1.1. Motivación del proyecto 3

1.2. Objetivos y enfoque 5

1.3. Metodología y plan de trabajo 7

2. Estado del arte **9**

2.1. Proyecto Hectospec 11

2.2. MEFOS 12

2.3. LAMOST 13

2.4. BigBOSS 13

3. Sistema, diseño y desarrollo **15**

3.1. Análisis de soluciones 17

3.1.1. Presentación de datos 18

3.1.2. Una placa de control para manejar todo el sistema 23

3.1.3. Una placa de control para manejar cinco secciones 24

3.1.4. Una placa de control para manejar dos secciones 26

3.1.5. Una placa de control para manejar cada sección 28

3.1.6. Resumen de soluciones 30

3.2. Diseño 31

3.2.1. Hardware 31

3.2.2. Software - Configuración del sistema 34

3.2.3. Software - Comunicación por parte del esclavo 36

3.3. Desarrollo 38

3.3.1. Hardware 38

3.3.2. Software 45

4. Experimentos Realizados y Resultados **49**

4.1. Escaneo del BUS I^2C en búsqueda de esclavos 52

4.2. Configuración de un canal en un MUX 54

4.3. Lectura de datos de un esclavo conectado al MUX 56

4.4. Cambio de MUX y canal 58

4.5. Escritura y Lectura de datos en un esclavo conectado a una celda 59

4.6. Escritura de datos en una celda completa 63

4.7. Resumen de pruebas 65

5. Conclusiones y trabajo futuro **67**

6. Bibliografía **71**

A. Artículo **73**

B. Presupuesto **87**

C. Pliego de condiciones **89**

Índice de figuras

1.1. NOAO 4-meter Mayall Telescope 4

2.1. Plano de posicionamiento 11

2.2. Posicionador 12

3.1. Actuador con fibra acoplada 17

3.2. Celda con 19 actuadores 19

3.3. Diagrama genérico de conexión 20

3.4. Trama de configuración del multiplexor 21

3.5. Diagrama de conexión para dos placas 24

3.6. Diagrama de conexión para cinco placas 26

3.7. Diagrama de conexión para diez placas 28

3.8. Distribución del plato focal 31

3.9. Diseño elegido para la conexión de las celdas al ordenador principal . . . 32

3.10. BUS I^2C 32

3.11. Tiempo de relleno 33

3.12. Trama de configuración para la recepción de datos 36

3.13. Beagle Bone 38

3.14. GPIO Pinout de Beagle Bone 39

3.15. Pinout de los multiplexores elegidos 40

3.16. Esquemático de la placa de comunicación 41

3.17. Diseño PCB de la placa de comunicación 42

3.18. Placa de comunicación 42

3.19. Pololu MinIMU-9 v3 42

3.20. Memoria 24C01 43

3.21. Adaptador de celda 1:19 43

3.22. Sistema Completo 44

4.1. Dirección de MUX PCA9548 52

4.2. Escaneo del BUS vacío 52

4.3. Escaneo del BUS con la placa de comunicación conectada 52

4.4. Análisis de trama con osciloscopio 53

4.5. Bits de selección de canal 54

4.6. Configuración de la salida del MUX y posterior escaneo 54

4.7. Tiempo de configuración del canal 4 en MUX 0x71 55

4.8. Consulta del registro de identificación del acelerómetro 56

4.9. Tiempo de consulta del registro de identificación 56

4.10. Consulta del registro de identificación del giroscópio 57

4.11. Tiempo de consulta del resitro de identificación 57

4.12. Código de selección de chip en memoria 24C01 58

4.13. Configuración de canal y detección de memorias 58

4.14. Escritura y lectura en memoria 0x50 59

4.15. Trama de escritura en memoria 0x50 59

4.16. Trama de lectura en memoria 0x50 60

4.17. Escritura de 14 coordenadas en memoria 61

4.18. Tiempo de espera entre el envío de cada byte 62

4.19. Configuración de una celda en su totalidad 63

Índice de cuadros

3.1. Ajuste de celdas 18

3.2. Potencia requerida por un switch según modelo 22

3.3. Potencia requerida por un MUX según modelo 22

3.4. Tiempos estimados en cada tarea 23

3.5. Tiempos estimados en cada tarea 25

3.6. Tiempos estimados en cada tarea 27

3.7. Tiempos estimados en cada tarea 29

3.8. Comparación de soluciones 30

3.9. Comparación de potencias en el plato focal 30

3.10. Tabla *SECTOR_INFO* 34

3.11. Tiempos estimados de lectura 37

4.1. Tiempos estimados en cada tarea 51

4.2. Tiempos estimados en cada tarea trabajando a 100KHz 65

4.3. Tiempos estimados en cada tarea trabajando a 400KHz 65

1

Introducción

- **Motivación del proyecto**
- **Objetivos y enfoque**
- **Metodología y plan de trabajo**

> Una de las principales enfermedades del hombre es su inquieta curiosidad por conocer lo que no puede llegar a saber.
>
> *Blaise Pascal*

El presente capítulo introductorio de la memoria
Expone las motivaciones para realizar el proyecto,
Así como los objetivos que pretende lograr
Y los pasos seguidos para tal propósito.

1.1. Motivación del proyecto

Tras incontables intentos por cartografiar el universo observable, la comunidad científica elaboró complejos mapas, detallando las propiedades de los cuerpos que podían observar en el firmamento.

Con el avance de la ciencia, los logros alcanzados en este campo han sido notables y constantemente mejorados. Surgiendo, durante los últimos años, multitud de proyectos cuyo objetivo principal ha sido dar respuesta a tantas cuestiones sin resolver, entre las que destacan la creación del universo, su expansión y el estudio de teorías entorno a la Energía y Materia Oscuras.

Uno de los proyectos más destacados en este campo es el SDSS-III (*Sloan Digital Sky Survey III*), con la participación del *Lawrence Berkeley National Laboratory* de Estados Unidos. Los principales objetivos de éste son, entre otros, el mapeo exhaustivo de la Vía Láctea, la búsqueda de planetas extrasolares y el estudio de la Energía Oscura.

Este proyecto tiene cuatro líneas principales de investigación:

- **SEGUE**: *Sloan Extension for Galactic Understanding and Exploration*, recopila el espectro de más de 230000 estrellas cercanas con el objetivo de estudiar la estructura interna de la Vía Láctea.
- **APOGEE**: *Apache Point Observatory Galactic Evolution Experiment*, estudia el interior de la galaxia y sigue Gigantes Rojas mediante un espectrógrafo de infrarrojos.
- **MARVELS**: *Multi-object APO Radial Velocity Exoplanet Large-area Survey*, estudia las características de los planetas extrasolares.
- **BOSS**: *Baryon Oscillation Spectroscopic Survey*, estudia la distribución espacial de las galaxias rojas y quásares[1], para detectar las características impresas por las oscilaciones acústicas de Baryon en el universo temprano.

Tras años de investigaciones y planteamiento de proyectos, la Universidad de Berkeley junto a diversas universidades y organismos tanto públicos como privados a lo largo del planeta, han puesto en marcha la creación del mayor mapa espectrográfico del universo observable bajo el nombre de *BigBOSS*. Éste, apoyándose en *BOSS*, toma como objetivo 50 millones de cuerpos y busca la localización más precisa hasta la fecha de, al menos, 20 millones de galaxias y quásares, para realizar un mapa con volumen de objetivos diez veces mayor al mejor mapa conseguido hasta la fecha.

[1] Un quásar, ***Quasi-Stellar** radio source*, es una fuente astronómica de energía electromagnética, que incluye radiofrecuencias y luz visible.

Este proyecto tiene como objetivo estudiar desde la Tierra el efecto de *Corrimiento al Rojo* en cientos de cuerpos, reduciendo así los costes de una misión espacial. Este efecto es la constatación del decaimiento frecuencial hacia el rojo. Con él, haciendo una analogía con el efecto *Doppler* en las ondas sonoras, es posible conocer con precisión el desplazamiento de objetos en el espacio, comparando la radiación reflejada por éstos con la procedente de una fuente emisora.

Para alcanzar los objetivos descritos, BigBOSS se servirá del *NOAO 4-meter Mayall Telescope* (*Kitt Peak, Arizona, EEUU*) sobre el que se montará un espectrógrafo que podrá estudiar en profundidad las propiedades del universo observable.

El diseño del sistema encargado de focalizar la radiación captada por el telescopio ha sido encargado a diferentes organismos, entre ellos a un conjunto de investigadores en España liderados desde el Instituto de Física Teórica (Campus de Excelencia UAM/CSIC). Este grupo ha confiado el diseño de la instrumentación al grupo de investigación HCTLab de la Escuela Politécnica Superior (EPS/UAM), participando también el Instituto Astrofísico de Andalucía (IAA/CSIC).

Y en este contexto se encuentra enmarcado el presente Proyecto Fin de Carrera.

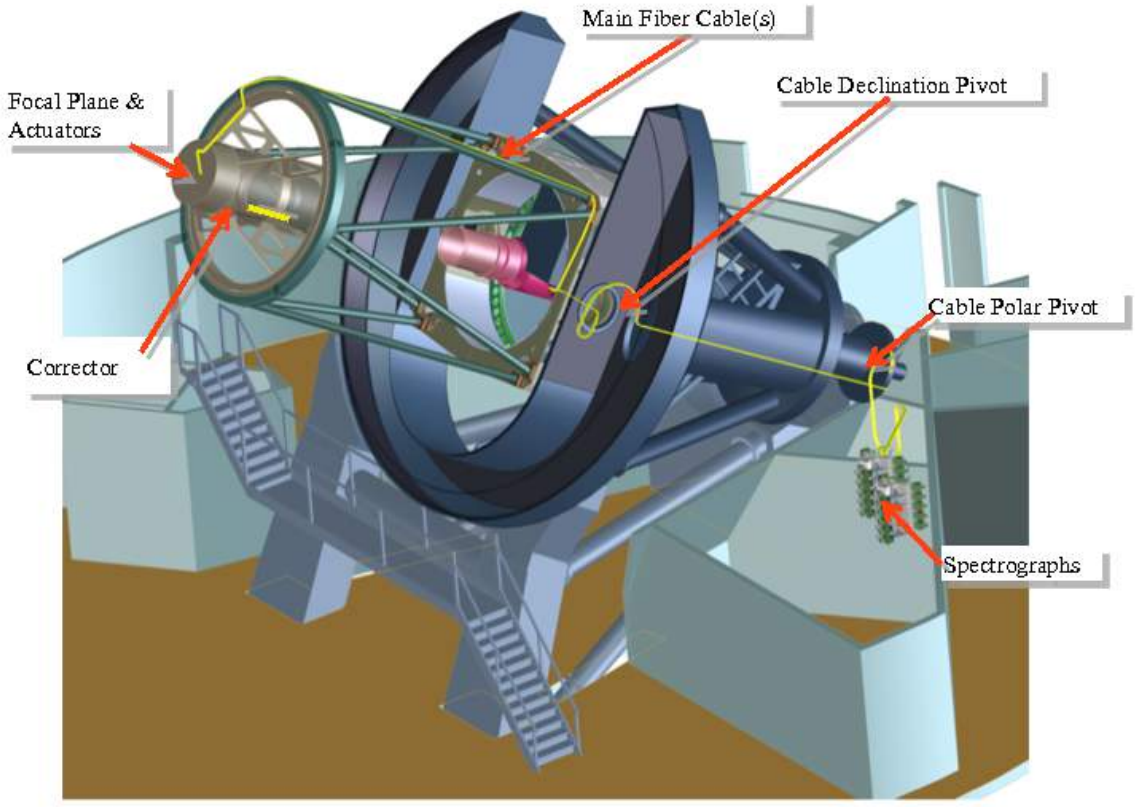

Figura 1.1: NOAO 4-meter Mayall Telescope

1.2. Objetivos y enfoque

Para cumplir su propósito, el proyecto BigBOSS propone utilizar cinco mil fibras ópticas situadas en el plano focal del espectrógrafo, cada una de ellas apuntando hacia una galaxia.

Una de las problemáticas del proyecto se centra en el posicionamiento de dichas fibras para que apunten en todo momento a la galaxia correspondiente. Esta tarea se compone, por una parte, de un dispositivo que se encarga de situar la fibra en una coordenada determinada, en adelante llamado actuador, y por otra parte en un sistema que transfiere dichas coordenadas desde una unidad central hacia dichos actuadores.

Como un actuador se encarga de posicionar una única fibra, habrá tantos actuadores como fibras y todos estarán ubicados en el plano focal que, en el proyecto BigBOSS, está dividido en diez secciones.

El objetivo del proyecto es el diseño y la construcción de un sistema capaz de proporcionar las coordenadas a los actuadores en un tiempo no superior a un segundo.

Una de las partes más importantes del proyecto es la toma de decisiones en cuanto a la arquitectura que se va a utilizar. Para ello se realizará una valoración previa de las ventajas e inconvenientes de las distintas soluciones posibles.

Para hallar la solución óptima, el proyecto se ha realizado en dos etapas. En la primera se realizó un estudio teórico en el que se hace comparación entre las distintas tecnologías disponibles, tanto alámbricas como inalámbrias. En la segunda, a la que se dedica este proyecto, se ha llevado a cabo la implementación práctica del estudio teórico anterior.

Para ello se estudiarán las siguientes posibles implementaciones:

- Utilizar una única placa de control, que se encargará de comunicar los datos a todos los actuadores.
- Utilizar varias placas de control, cada una encargándose de unas cuantas secciones de las diez que forman el disco.
    - Una placa de control gestiona cinco secciones
    - Una placa de control gestiona dos secciones
    - Una placa de control gestiona una sección

Este estudio verifica la viabilidad de cada solución, así como el cumplimiento de algunas restricciones como:

- **Velocidad:** A pesar de que se toma una imagen cada 30 minutos, el sistema debe ser capaz de enviar las coordenadas a cada actuador en un tiempo muy limitado, de esta forma se aprovechará el resto de tiempo para chequear posibles errores en la nube de los actuadores.
- **Energía:** Puesto que el sistema de posicionamiento se ubica sobre el plato focal, debe tener un consumo lo mínimo posible para no calentar el resto de la estructura.
- **Comunicación:** Además de que el sistema permita enviar las coordenadas desde el ordenador principal hacia los actuadores, estos también deben ser capaces de enviar algún tipo de señal de alerta en el caso de que sea necesario.

1.3. Metodología y plan de trabajo

Se ha seguido la siguiente organización de trabajo para la realización del proyecto:

- **Formación previa y estudio del estado del arte**

  En primer lugar se han adquirido los conocimientos necesarios para abordar el proyecto. Se ha estudiado el funcionamiento básico de los componentes involucrados en el desarrollo del sistema de comunicación.

  De manera simultánea se ha estudiado el estado del arte, investigando los avances que se producen en el campo que utiliza esta tecnología.

- **Decisión de la arquitectura a utilizar**

  Tras una comparación de tiempo y energía necesarios para cada una de las arquitecturas propuestas, se ha decidido la que mejor se adapta a las necesidades del proyecto.

- **Desarrollo e implementación**

  Se ha procedido a fabricar el sistema propuesto anteriormente. No se ha construido el sistema completo para los cinco mil actuadores sino un sistema mínimo que permite validar la arquitectura propuesta. También se ha desarrollado el paquete software necesario para su utilización.

- **Prueba de resultados y evaluación**

  Se han realizado las pruebas necesarias para comprobar si el funcionamiento es el esperado.

- **Conclusiones y Memoria**

  Por último se ha realizado esta memoria utilizando los datos recopilados a lo largo del proyecto.

2

Estado del arte

Es un hecho que el hombre tiene que controlar la ciencia y chequear ocasionalmente el avance de la tecnología.

Thomas Henry Huxley

El presente capítulo pretende mostrar las tecnologías actuales
Y el contexto en el que se ha realizado este trabajo.

Durante los últimos años, en el campo de la astrofísica, se han desarrollado multitud de proyectos basados en espectrógrafos, cuya característica principal ha sido el posicionamiento de fibras ópticas en el plano focal de telescopios con el fin de focalizar la radiación extra-planetaria.

El objetivo principal de estos proyectos ha sido transformar telescopios con resoluciones moderadamente aceptables en sistemas capaces de analizar simultáneamente millares de objetos y estudiar sus propiedades en profundidad.

En una primera etapa, estos sistemas basaron su funcionamiento en el posicionamiento estático de las fibras por medio de anclajes fijos. Más adelante se desarrollaron prototipos *Pick and Place* basados en el enganche y suelte de las fibras dentro de una superficie pre-establecida (pegamentos o taladros), mediante procesos manuales o robotizados. En los desarrollos más modernos se están utilizando posicionadores independientes cuyos principios, grados de libertad, coste y funcionalidad varían significativamente de unos a otros.

2.1. Proyecto Hectospec

El proyecto Hectospec puede ser considerado uno de los exponentes del método Pick and Place robotizado.

Al igual que en BigBoss, la herramienta principal es un espectrógrafo de resolución moderada basado en la recepción de radiación mediante fibras ópticas.

Este instrumento se sirve del método pick-and-place para posicionar un total de 300 cabezales de fibra en el plano focal del telescopio NMT (La Silla, Chile).

El tiempo total de posicionamiento de las fibras es de 300 segundos, con una precisión de 25um.

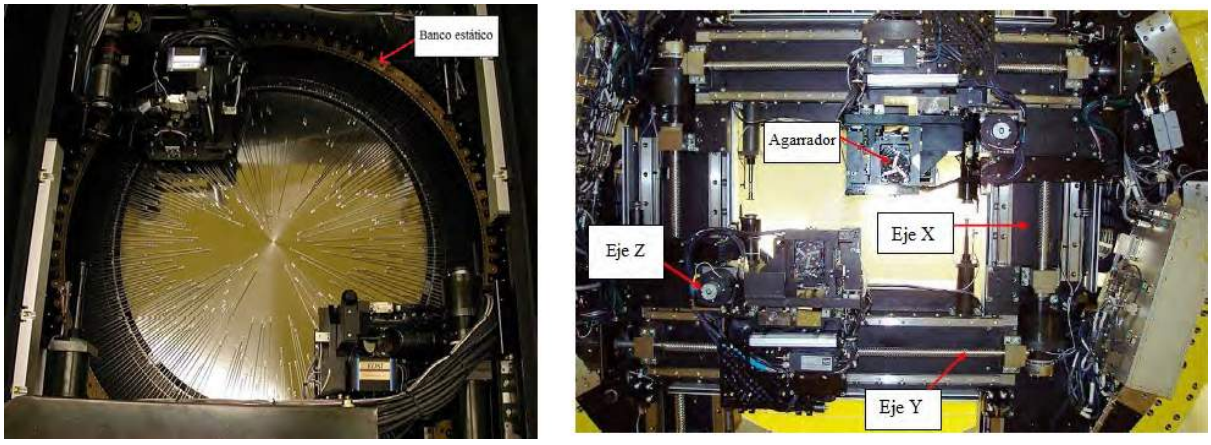

Figura 2.1: Plano de posicionamiento

2.2. MEFOS

Con el objetivo de disminuir el tiempo empleado en el posicionamiento secuencial de fibras en proyectos como el Hectospec, se diseñaron sistemas robotizados que pudiesen trabajar en paralelo (sacrificando la resolución).

Uno de los primeros proyectos que materializó este tipo de sistemas fue el MEFOS[1], que utilizó un conjunto de brazos, instalados alrededor de la superficie de posicionado, cuyo único grado de libertad les permite rotar un determinado número de grados en su base.

En este proyecto se posicionan 30 fibras en un tiempo de 240 segundos

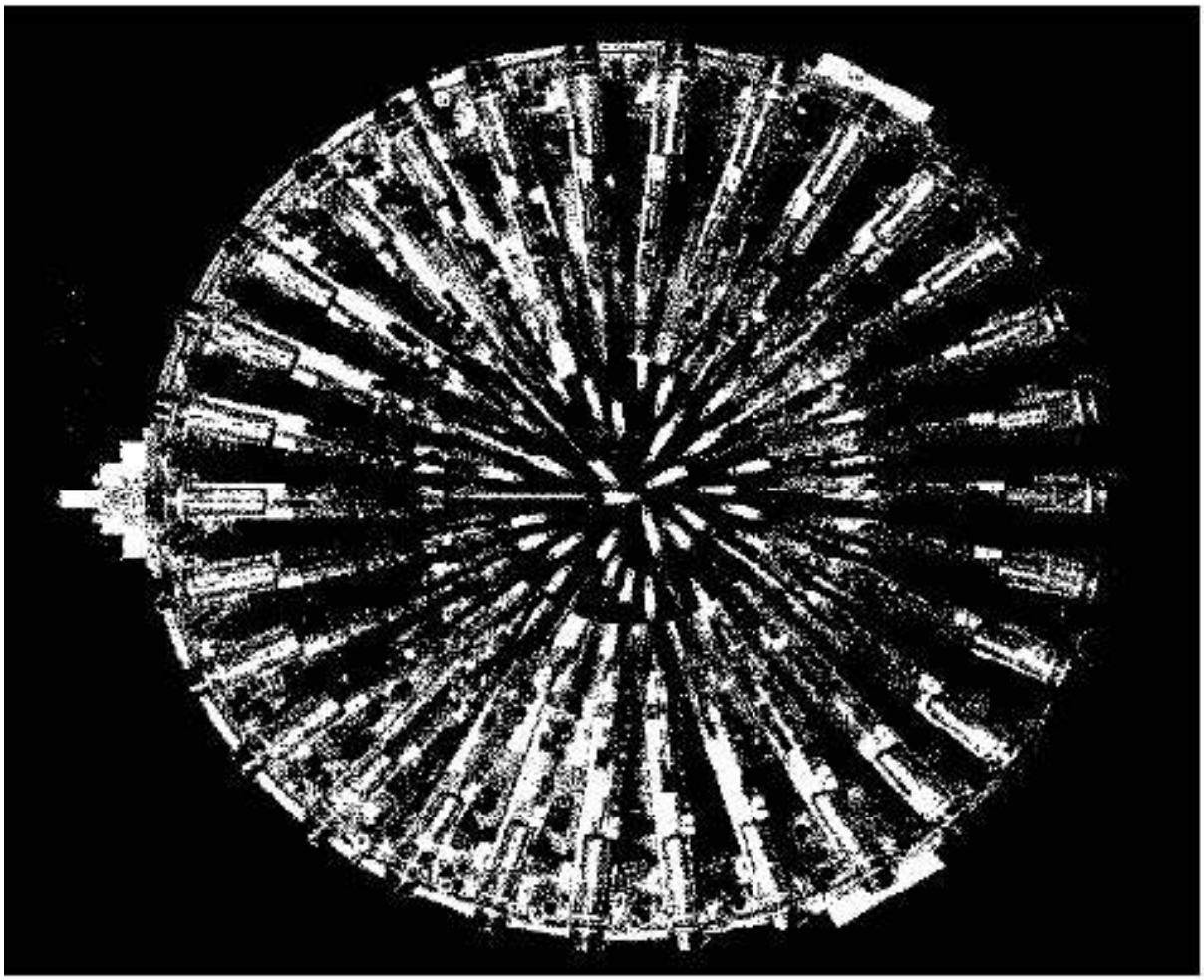

Figura 2.2: Posicionador

[1]Meudon-ESO Fiber Optical System

2.3. LAMOST

El proyecto chino LAMOST[2] dirigido por la Academia China de Ciencias, busca estudiar en profundidad las propiedades de la energía oscura, recopilando el espectro de cuerpos galácticos y extra galácticos.

Para ello se sirve de un telescopio instalado en el *Xianglong Station of national Astronomical Observatory*. Este telescopio consta de dos espejos reflectores masivos que focalizan la radiación captada hacia una superficie de exposición de 1.75m, formada por 4000 actuadores independientes.

El proyecto LAMOST es capaz de posicionar sus 4000 actuadores con una precisión de 40um en un tiempo de 600 segundos.

2.4. BigBOSS

El sistema está compuesto por 5000 actuadores independientes agrupados en celdas hexagonales. En el año 2013 se hizo una propuesta de la electrónica de control para este proyecto, llevada a cabo en el proyecto de fin de carrera de Nasib Fahim Fernández[3].

Cada actuador está formado por dos motores que colocan cada fibra en su punto de exposición de forma coplanar, esto es, ambos motores hacen mover la fibra en el mismo plano, con dos grados de libertad, de forma angular.

El sistema diseñado permite posicionar los 5000 actuadores con una precisión de 5um en un tiempo de 60s.

En el mismo año Jesús Castro Murillas realizó un estudio teórico[4] sobre cuál sería la forma de implementar una comunicación entre los actuadores y el ordenador central, llegando a la conclusión que la opción óptima es combinar una solución Ethernet e I^2C.

[2]Large sky area Multi-object fibre Spectrocopy Telescope
[3]Ver [1] en Bibliografía
[4]Ver [2] en Bibliografía

3

Sistema, diseño y desarrollo

- Análisis de soluciones
- Diseño
- Desarrollo

> Es de importancia para quien desee alcanzar una certeza en su investigación, el saber dudar a tiempo.
>
> *Aristóteles*

Este capítulo presenta las bases teóricas de partida,
Los estudios hechos sobre las mismas
Y el diseño final elegido.

También se expone el desarrollo de la solución final.

3.1. Análisis de soluciones

Tal como se comentó anteriormente, la problemática que se plantea es la de posicionar las cinco mil fibras, agrupadas en celdas hexagonales, situadas en el plano focal del espectrógrafo, tarea que se compone por el propio posicionamiento de la fibra mediante un actuador y por un sistema que transfiere las coordenadas de un ordenador central a los actuadores, siendo esta última parte el objetivo de este proyecto.

Durante la primera etapa del proyecto, realizada por Jesús Castro Murillas[1], se llegó a la conclusión de que la manera óptima de implementar la solución es mediante una topología de tipo árbol junto con una arquitectura de tipo bus para comunicarse con los actuadores de una celda.

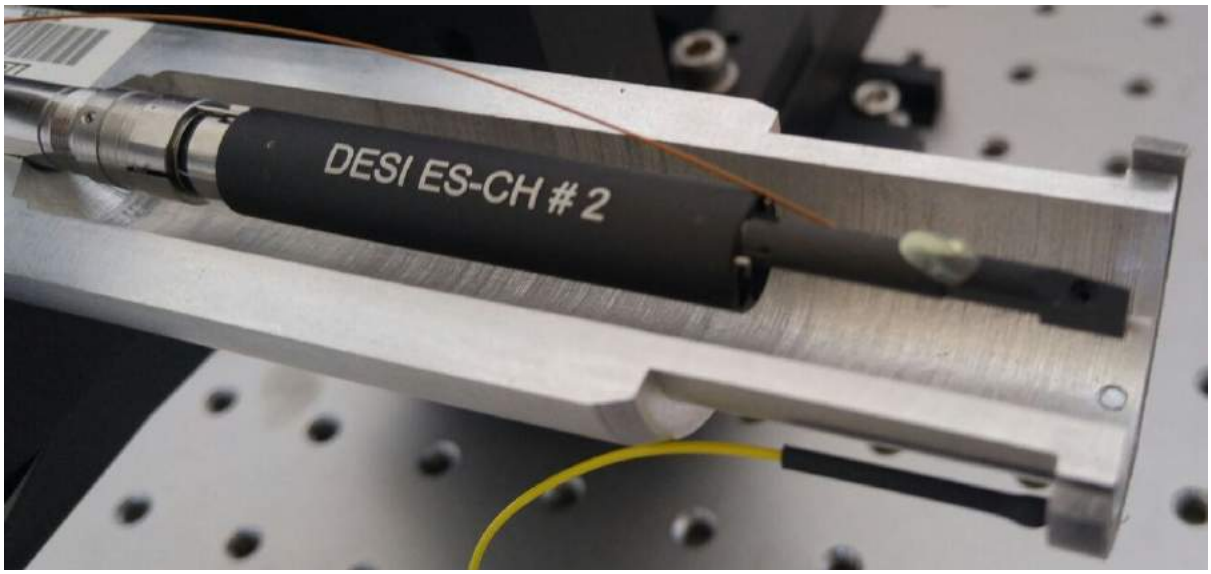

Figura 3.1: Actuador con fibra acoplada

En concreto se utilizará una comunicación *Ethernet*, para enlazar el ordenador central con la placa de control, y una comunicación I^2C, para enlazar la placa de control con los actuadores situados en las celdas.

[1] Ver [2] en Bibliografía

3.1.1. Presentación de datos

Como se comentó, se plantea enviar las coordenadas de las fibras desde un ordenador central hasta los actuadores encargador de posicionar dichas fibras. Esto se hará mediante una placa de control que conectará ambos extremos.

Esta placa tiene un puerto de comunicaciones I^2C, que irá derivándose mediante multiplexores para cubrir todos los sectores del plato focal que contienen los actuadores con las fibras.

Para tal propósito, se va a estudiar la viabilidad de utilizar tan solo una placa de control, o la necesidad de utilizar más.

Recordar que el plato focal está formado por diez secciones y cada una de estas contiene las celdas con los actuadores. Teniendo esto en cuenta, y que el número de fibras totales es aproximadamente cinco mil, obtenemos que cada sección debe tener unas 500 fibras.

También hay que tener en cuenta la agrupación de las celdas, que ha de tener una forma hexagonal. Con estas premisas se proponen las siguientes agrupaciones:

FIBRAS POR CELDA	CELDAS POR SECCIÓN	FIBRAS POR SECCIÓN
37	14	518
19	27	513
7	72	504

Cuadro 3.1: Ajuste de celdas

Por comodidad de trabajo se toma una cantidad de 19 fibras por celda, lo que implica 27 celdas por sección y un total de 513 fibras en la misma.

No se escoge el valor superior pues habría una superpoblación de fibras por celda, y tampoco se toma el valor inferior pues habría una superpoblación de celdas por sección.

En la siguiente imagen se puede observar una maqueta en 3D que representa una celda con los 19 actuadores correspondientes.

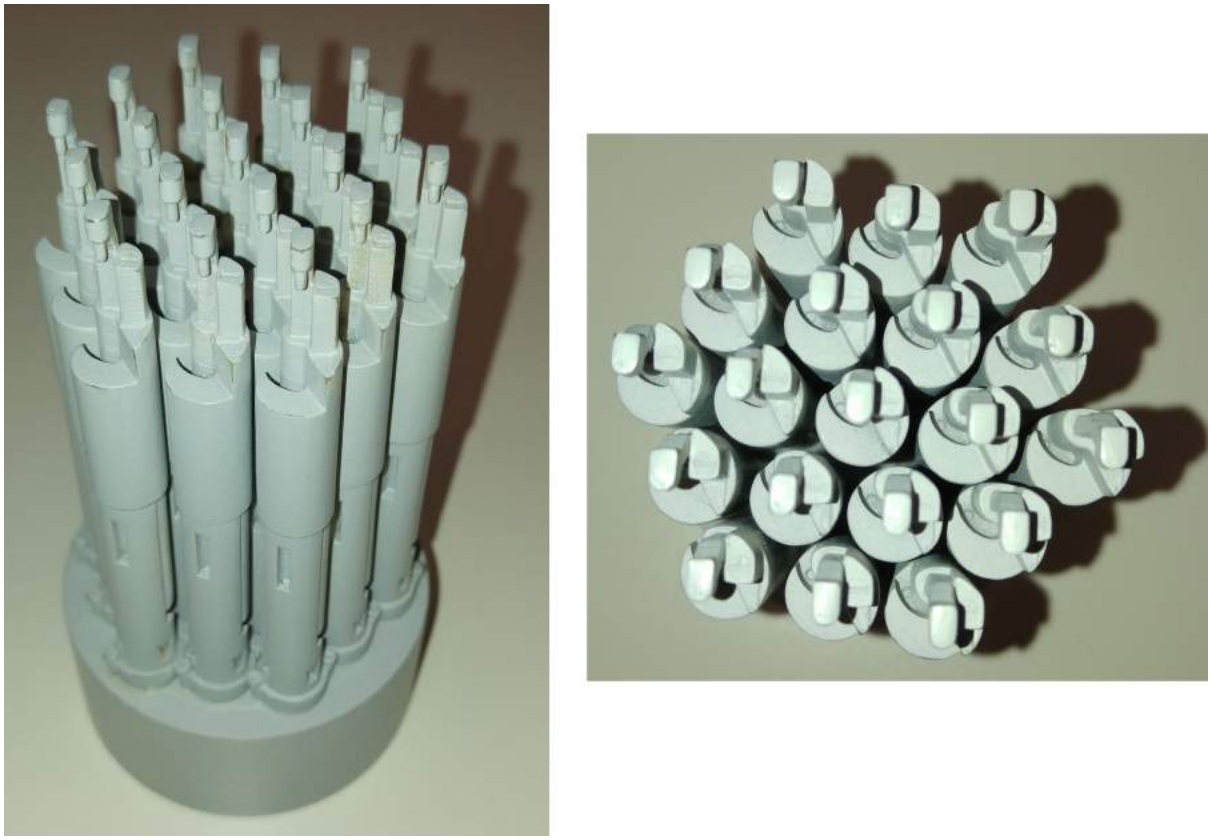

Figura 3.2: Celda con 19 actuadores

Una vez decidido el número de fibras por celda, y sabiendo el número de fibras totales por sección, se presenta el siguiente diagrama de un caso particular de distribución de los elementos que componen el sistema propuesto.

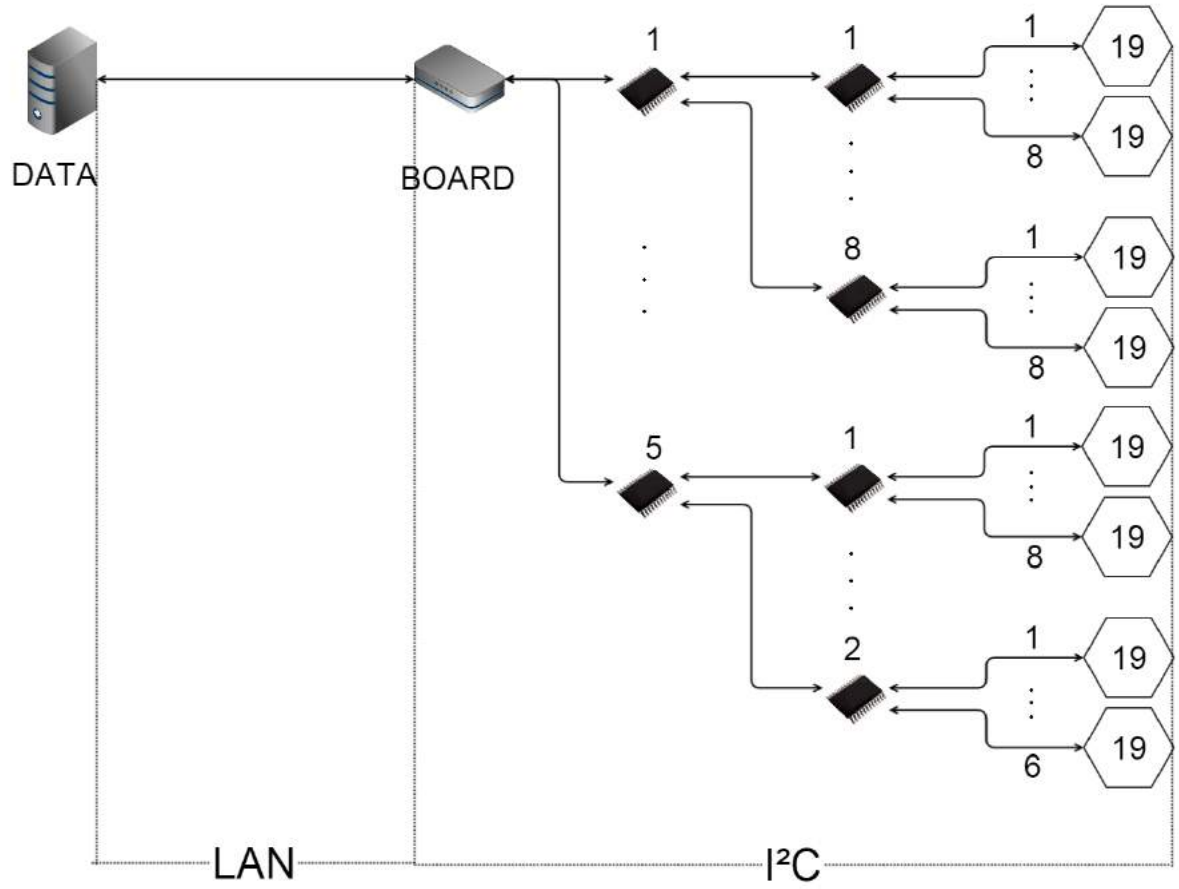

Figura 3.3: Diagrama genérico de conexión

Ahora que se ha presentado un modelo de conexión de los componentes, se procede a realizar las estimaciones para el distinto número de placas de control. En este caso se tendrán en cuenta cuatro opciones:

- Una sola placa controla las diez secciones, 5130 actuadores a lo sumo
- Una placa controla cinco secciones, 2565 actuadores a lo sumo
- Una placa controla dos secciones, 1026 actuadores a lo sumo
- Una placa por sección, 513 actuadores a lo sumo

Además de estos casos hay que tener en cuenta otros datos para los cálculos:

- La velocidad de transmisión Ethernet con la que se trabajará es de 1 Gbps.
- La velocidad de transmisión I^2C con la que se pretende trabajar es de 400 Kbps o 1 Mbps.
- Cada actuador requiere dos coordenadas para posicionar la fibra que tiene acoplada. Además, en el proyecto de Nasib Fahim Fernández[2], se definió un algoritmo de siete pasos que pretende solucionar la problemática de colisión entre los actuadores que se encuentran situados muy próximos uno del otro. Cada coordenada se representa mediante un byte. Todo esto implica que se transmitirán $(2 \times 7) \times 8$ bits de datos crudos por actuador. Como en cada sección hay 513 actuadores, esto implica que cada sección requiere 57456 bit de datos.
- También hay que tener en cuenta las tramas que sirven para manejar los multiplexores:
  - 8+1 bit para seleccionar el multiplexor deseado
  - 8+1 bit para seleccionar la salida dentro del multiplexor

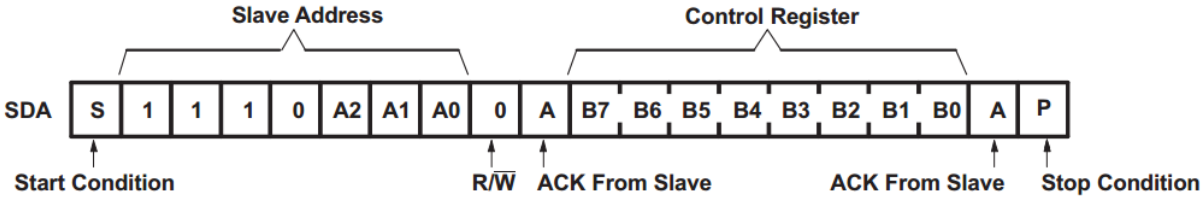

Figura 3.4: Trama de configuración del multiplexor

- Cada multiplexor de la última etapa puede manejar a lo sumo 152 actuadores (19×8).
- Cada comunicación I^2C de tipo escritura requiere un ACK[3], resultando en 14 bit de datos más por cada actuador.

[2]Ver [1] en Bibliografía

[3]Del inglés *acknowledgement*, acuse de recibo o asentimiento. Es un mensaje que el destino envía al origen para confirmar la recepción del mensaje.

Además de los bits de datos necesarios, hay que tener en cuenta el consumo que genera el sistema.

- Según el manual de la placa de control elegida, su consumo es aproximadamente de 450mA @ 5V, lo que en términos de potencia implica un consumo de 2.25W.
- Comparando el consumo de varios switch disponibles en el mercado, se ha obtenido la siguiente tabla con los valores medios.

# PUERTOS GIGABIT	POTENCIA (W)
5	3
8	5
16	13

Cuadro 3.2: Potencia requerida por un switch según modelo

- Por último, hay que tener en cuenta el consumo de los MUX[4]. Según las correspondientes hojas de características, se presenta la siguiente tabla.

F (KHz)	uA	Potencia @ 3.3V (uW)
400	50	165
1000	150	495

Cuadro 3.3: Potencia requerida por un MUX según modelo

Una vez presentados estos datos, se procede con cada caso en particular.

[4]Se utiliza MUX como abreviación de Multiplexor.

3.1.2. Una placa de control para manejar todo el sistema

En este caso, el diagrama que se aplica es el correspondiente a la Figura 3.3.

El primer paso, antes de poder enviar los datos al actuador, es configurar el camino deseado a través de los MUX. Para ello se necesitan 8 bit, con el correspondiente ACK, para la selección del MUX y otros 8, con su correspondiente ACK, para la selección de la salida que nos interesa dentro del MUX. El procedimiento para enviar los datos a los actuadores es el siguiente:

- Selección del MUX de primera etapa (9 bit)
  - Selección de canal en el MUX de primera etapa (9 bit)
    - Selección del MUX de segunda etapa (9 bit)
      - Selección de canal en el MUX de segunda etapa (9 bit)
      - Envío de datos a los 19 actuadores $19 \times (112 + 14)$ bit

Esta configuración está formada por 39 multiplexores de ocho canales, utilizando un total de 309 salidas de las 312 disponibles.

Teniendo esto en cuenta, se puede construir la siguiente tabla:

Tarea	**bits**	**Velocidad**	**Tiempo(s)**	**Comentarios**
Envío de datos PC -> Placa	574560	1Gbps	574×10^{-6}	Se reciben los datos del PC
Configuración MUX	351	400Kbps 1Mbps	878×10^{-6} 351×10^{-6}	Tiempo de configuración de todos los MUX
Configuración Canal	2781	400Kbps 1Mbps	6953×10^{-6} 2781×10^{-6}	Tiempo de configuración de las salidas de MUX
Envío de datos Placa -> Actuadores	646380	400Kbps 1Mbps	1,616 0,646	Tiempo de envío de datos a los actuadores + ACK

Cuadro 3.4: Tiempos estimados en cada tarea

Teniendo en cuenta la tabla presentada, el tiempo total utilizado por esta configuración es de 1.63s@400KHz o 0.65s@1MHz.

El tiempo total consumido en configurar un actuador, suponiendo que los datos ya se encuentran en la placa de control, es de 405us@400KHz o de 162us@1MHz. Esta suposición se hace puesto que el tiempo de transferencia de datos, para un único actuador, entre el PC y la placa de control es despreciable frente al resto de tiempos que se manejan.

Recurriendo al Cuadro 3.2 y Cuadro 3.3 se obtiene que se requiere 6.44mW@400KHz o 19.31mW@1MHz para alimentar los 39 MUX-8 y 2.25W para la placa, resultando en total un consumo máximo de 2.27W.

3.1.3. Una placa de control para manejar cinco secciones

Recordemos que para manejar una sección hacen falta tres MUX-8 completos y tres salidas de otro MUX-8. Puesto que ahora se tienen dos placas de control, el diseño presentado sufre un ligero cambio.

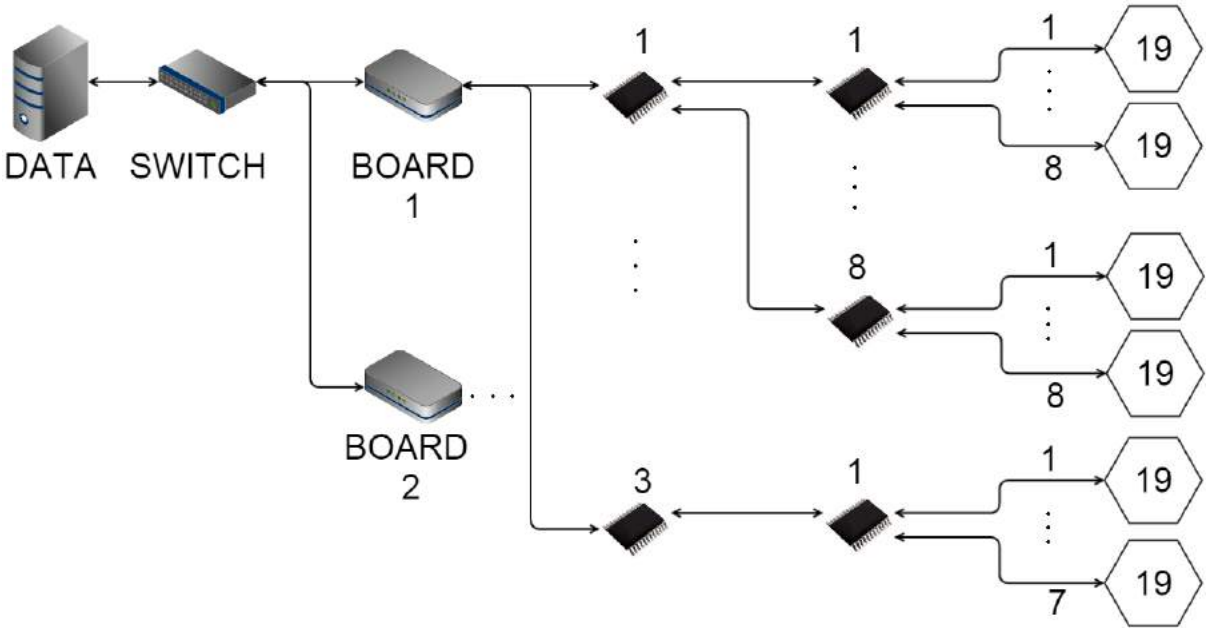

Figura 3.5: Diagrama de conexión para dos placas

Como se puede observar, en la primera etapa ahora se utilizan tres MUX-8 en lugar de cinco, y del último se usa sólo una salida.

Dado que el número de etapas no ha variado, se puede mantener el algoritmo:

- Selección del MUX de primera etapa (9 bit)
  - Selección de canal en el MUX de primera etapa (9 bit)
    - Selección del MUX de segunda etapa (9 bit)
      - Selección de canal en el MUX de segunda etapa (9 bit)
      - Envío de datos a los 19 actuadores $19 \times (112 + 14)$ bit

En esta configuración cada placa da uso de 20 MUX-8 y se utilizan 155 salidas.

A continuación se presenta la tabla de tiempos con los valores actualizados:

Tarea	**bits**	**Velocidad**	**Tiempo(s)**	**Comentarios**
Envío de datos PC -> Placa	287280	1Gbps	287×10^{-6}	Se reciben los datos del PC
Configuración MUX	180	400Kbps 1Mbps	450×10^{-6} 180×10^{-6}	Tiempo de configuración de todos los MUX
Configuración Canal	1395	400Kbps 1Mbps	3488×10^{-6} 1395×10^{-6}	Tiempo de configuración de las salidas de MUX
Envío de datos Placa -> Actuadores	323190	400Kbps 1Mbps	0,808 0,323	Tiempo de envío de datos a los actuadores + ACK

Cuadro 3.5: Tiempos estimados en cada tarea

El tiempo total utilizado por esta configuración es de 0.82s@400KHz o 0.33@1MHz.

En términos de consumo, se requieren 6.6mW@400KHz o 19.8mW@1MHz para alimentar los 40 MUX-8; 4.5W para las dos placas de control y 3W para el switch de cinco puertos, resultando en un consumo total de como máximo 7.52W.

Puesto que el algoritmo no cambia, el tiempo de configuración de un único actuador tampoco cambia.

3.1.4. Una placa de control para manejar dos secciones

Puesto que una sección necesita de tres MUX-8 y tres salidas de un MUX-8 más, nos hará falta sólo una etapa de MUX. Esto implica un cambio importante tanto en el circuito como en los datos que se tienen en cuenta para realizar los cálculos.

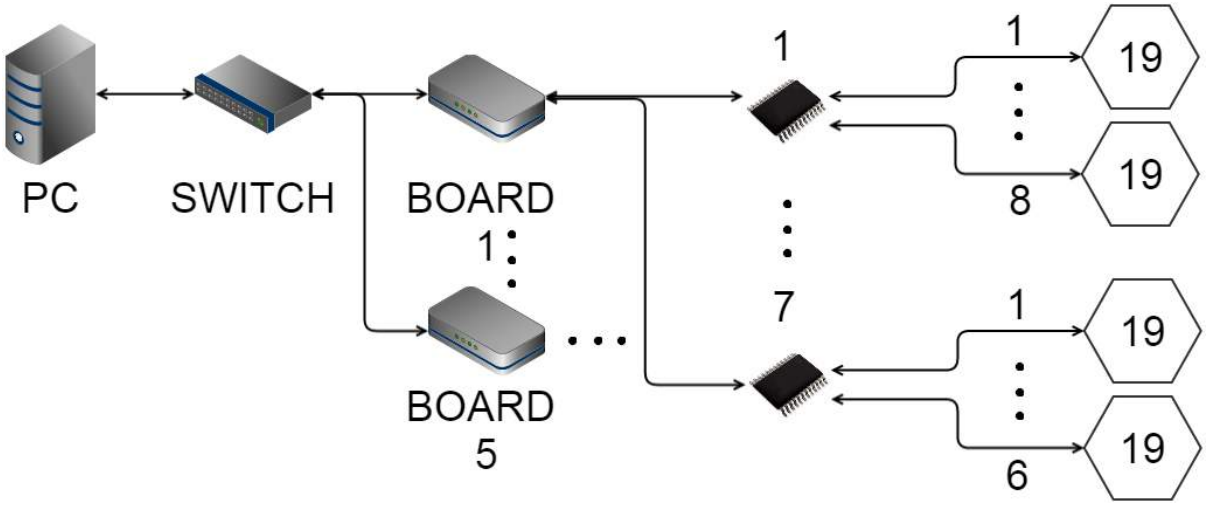

Figura 3.6: Diagrama de conexión para cinco placas

Puesto que hay una etapa menos, el nuevo algoritmo es el siguiente:

- Selección del MUX (9 bit)
  - Selección de canal en el MUX (9 bit)
    - Envío de datos a los 19 actuadores $19 \times (112 + 14)$ bit

Esta configuración requiere siete multiplexores de ocho canales, y en total se utilizan 54 canales de los 56 disponibles.

Actualizando la tabla de los tiempos obtenemos el siguiente resultado:

Tarea	**bits**	**Velocidad**	**Tiempo(s)**	**Comentarios**
Envío de datos PC -> Placa	114912	1Gbps	114×10^{-6}	Se reciben los datos del PC
Configuración MUX	63	400Kbps 1Mbps	158×10^{-6} 63×10^{-6}	Tiempo de configuración de todos los MUX
Configuración Canal	549	400Kbps 1Mbps	1372×10^{-6} 549×10^{-6}	Tiempo de configuración de las salidas de MUX
Envío de datos Placa -> Actuadores	129276	400Kbps 1Mbps	0,323 0,129	Tiempo de envío de datos a los actuadores + ACK

Cuadro 3.6: Tiempos estimados en cada tarea

El tiempo total utilizado por esta configuración es de 0.33s@400KHz o 0.13s@1MHz.

El tiempo total consumido en configurar un actuador, suponiendo que los datos ya se encuentran en la placa de control, es de 360us@400KHz o de 144us@1MHz. Esta suposición se hace puesto que el tiempo de transferencia de datos, para un único actuador, entre el PC y la placa de control es despreciable frente al resto de tiempos que se manejan.

En términos de consumo, se requieren 5.78mW@400KHz o 17.33mW@1MHz para alimentar los 35 multiplexores, 11.25W para las cinco placas de control y 5W para el switch, resultando un consumo total de momo máximo 16.27W.

3.1.5. Una placa de control para manejar cada sección

Recordemos que cada sección es manejada por tres MUX-8 y tres salidas de un MUX-8 adicional de la última etapa, por tanto el diagrama de conexión no sufre grandes cambios respecto al caso anterior.

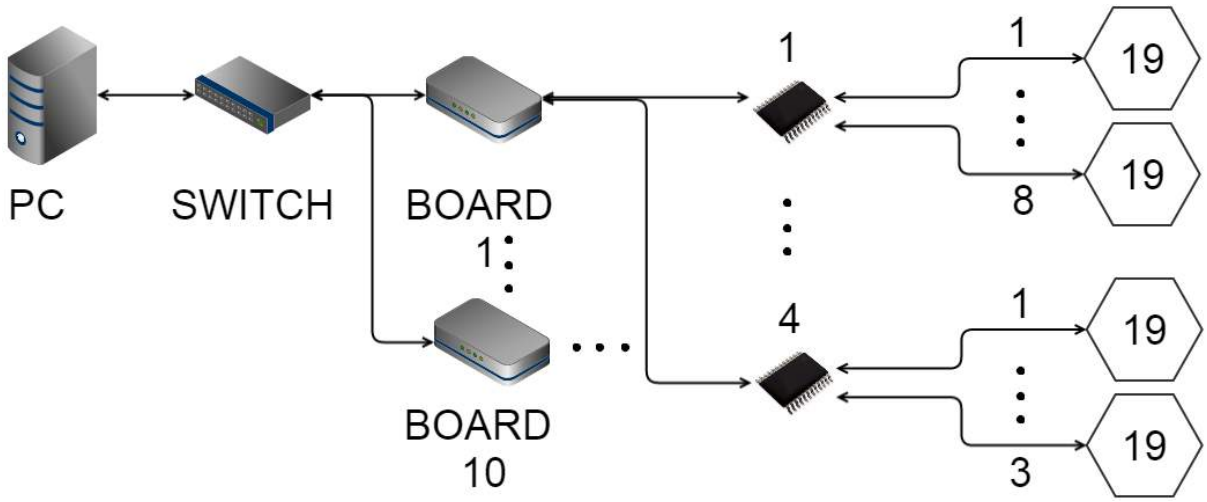

Figura 3.7: Diagrama de conexión para diez placas

Ya que el número de etapas no ha cambiado, el algoritmo de configuración de los actuadores se mantiene:

- Selección del MUX (9 bit)
  - Selección de canal en el MUX (9 bit)
    - Envío de datos a los 19 actuadores $19 \times (112 + 14)$ bit

En esta configuración, cada placa da uso de cuatro MUX-8 y en total se utilizan 27 canales.

Actualizando la tabla de los tiempos con los nuevos valores obtenemos:

Tarea	**bits**	**Velocidad**	**Tiempo(s)**	**Comentarios**
Envío de datos PC -> Placa	57456	1Gbps	57×10^{-6}	Se reciben los datos del PC
Configuración MUX	36	400Kbps 1Mbps	90×10^{-6} 36×10^{-6}	Tiempo de configuración de todos los MUX
Configuración Canal	243	400Kbps 1Mbps	608×10^{-6} 243×10^{-6}	Tiempo de configuración de las salidas de MUX
Envío de datos Placa -> Actuadores	64638	400Kbps 1Mbps	0,162 0,065	Tiempo de envío de datos a los actuadores + ACK

Cuadro 3.7: Tiempos estimados en cada tarea

El tiempo total utilizado por esta configuración es de 0.16s@400KHz o 0.07s@1MHz.

En términos de consumo, se requieren 6.6mW@400KHz o 19.8mW@1MHz para alimentar los cuarenta multiplexores, 22.5W para las diez placas de control y 13W para el switch de 16 puertos, resultando un consumo total de como máximo 35.52W.

Puesto que el algoritmo no cambia, el tiempo de configuración de un único actuador tampoco cambia.

3.1.6. Resumen de soluciones

A modo de resumen de las soluciones planteadas en los apartados anteriores, se presenta la siguiente tabla comparativa:

$\#Placas$	$\#MUX-8$	$Switch$	$T_{400KHz}(s)$	$T_{1MHz}(s)$	$P(W)$	$E_{400KHz}(J)$	$E_{1MHz}(J)$
1	39	-	1.63	0.65	2.27	3.71	1.48
2	40	5 Gigabit	0.82	0.33	7.52	6.17	2.48
5	35	8 Gigabit	0.37	0.15	16.27	6.02	2.44
10	40	16 Gigabit	0.16	0.07	35.52	5.68	2.49

Cuadro 3.8: Comparación de soluciones

También es interesante conocer la potencia que hay en el plato focal:

$\#Placas$	$\#MUX-8$	$P_{Total}(W)$	$P_{Plato}(W)$	$P_{Exterior}(W)$
1	39	2.27	2.27	0
2	40	7.52	4.52	3
5	35	16.27	11.27	5
10	40	35.52	22.52	13

Cuadro 3.9: Comparación de potencias en el plato focal

Puede verse que la opción óptima en cuanto al consumo total y coste del sistema es la de utilizar una placa de control y multiplexores de 1MHz para controlar todo el sistema, sacrificando para ello el tiempo de configuración y posible estabilidad del sistema.

La opción óptima en cuanto al tiempo de configuración es la de utilizar una placa de control por sección, junto con multiplexores de 1MHz, sacrificando para ello la energía consumida durante el proceso y, lo que es más notorio, el coste del sistema, puesto que se utilizan diez veces más placas de control que en la primera opción, además de un switch.

En cuanto a la estabilidad del sistema, la opción óptima es la de utilizar multiplexores de 400KHz, sacrificando para ello la energía y el tiempo de configuración.

3.2. Diseño

3.2.1. Hardware

Ante la necesidad de cumplir con unos tiempos críticos para el envío de coordenadas a los actuadores, se decide utilizar la solución de una placa de control por cada sección del plato focal, de esta forma se obtienen los tiempos óptimos, gracias a la paralelización de las tareas.

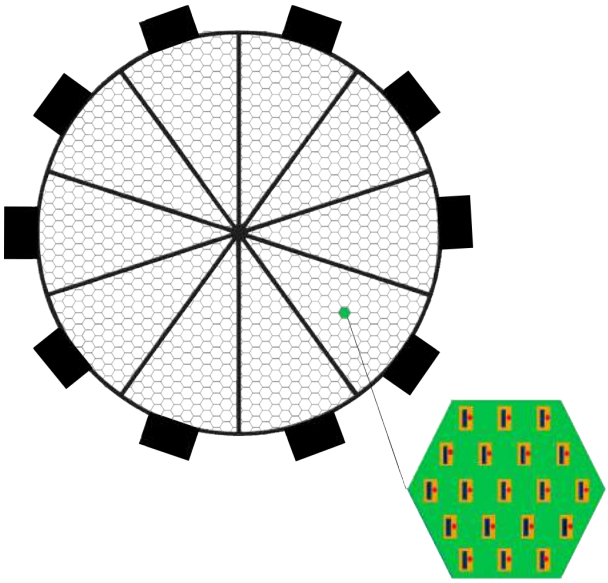

Figura 3.8: Distribución del plato focal

En la figura se puede ver que el plato focal está dividido en diez secciones, cada una gobernada por una placa de control.

Tal como se introdujo en el apartado *3.1.1.* cada sección contiene 27 celdas y cada celda está compuesta por 19 actuadores, por lo que cada sección tiene a lo sumo 513 actuadores.

Cada placa de control estará conectada mediante ethernet a un switch, que a su vez será el encargado de comunicar las diez placas de control con el ordenador central. La comunicación entre cada placa de control y los 513 actuadores de su sección será mediante una red de multiplexores I^2C.

Dicha red está compuesta por cuatro multiplexores, siendo estos capaces de manejar hasta 32 celdas. Puesto que tan solo se necesita cubrir 27 celdas, se procede a realizar un balanceo de canales proporcionados. Esto es, en vez de cargar tres multiplexores por completo y utilizar tres salidas del cuarto se utilizarán siete de los ocho canales en tres multiplexores y seis de los ocho canales del cuarto, cubriendo así las 27 celdas de una manera más proporcional.

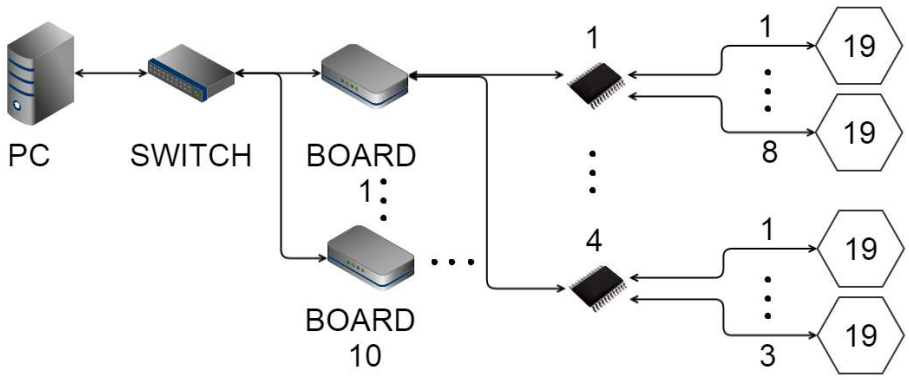

Figura 3.9: Diseño elegido para la conexión de las celdas al ordenador principal

Recordar que cada celda está formada por 19 actuadores, que irán conectados a una de las salidas del multiplexor.

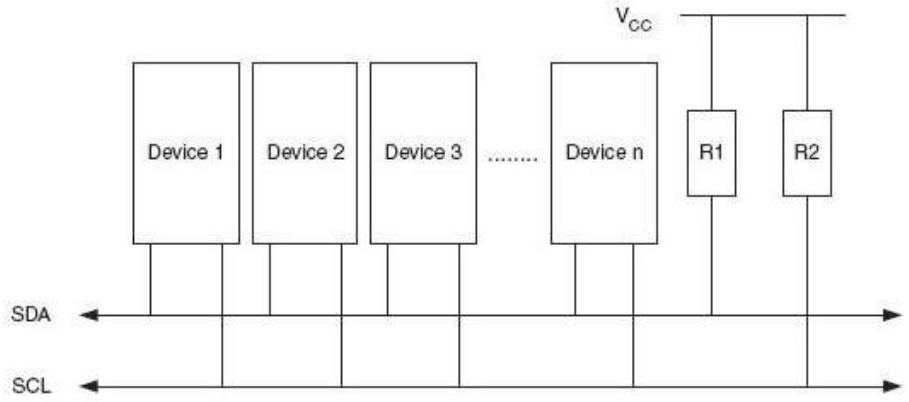

Figura 3.10: BUS I^2C

En resumen, este diseño permite acelerar la carga de los datos en los actuadores gracias a que se paraleliza el trabajo, ya que cada sección posee un control independiente de las demás.

Recuperando el algoritmo presentado en la sección *3.1.4.*, se puede ilustrar el tiempo que tomaría en rellenar una sección al completo con el sistema propuesto.

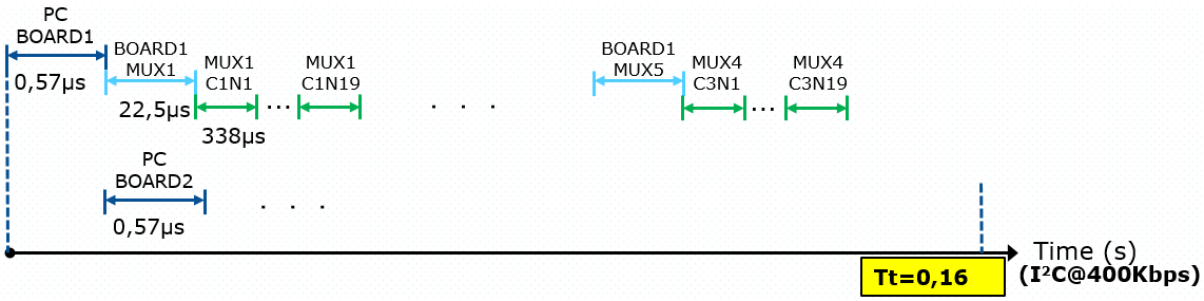

Figura 3.11: Tiempo de relleno

De una manera más detallada, los pasos en cada placa de control son los siguientes:

- Carga desde el PC de todas las coordenadas de la sección
- Selección del MUX
  - Selección de canal en el MUX
    - Envío de datos a cada uno de los 19 actuadores

3.2.2. Software - Configuración del sistema

El paquete software se ha dividido en dos partes:

- El **núcleo** del programa reside en cada una de las placas de control, haciéndolas así independientes entre ellas. Esta parte se ha desarrollado íntegramente en *C*.
- El *módulo de gestión de las coordenadas* se ubica en el ordenador principal. La implementación se ha llevado a cabo en *Python*.

Empezando por la segunda parte, la estructura de la base de datos es la siguiente:

ID	SECTOR_ID	FIBER_ID	X_POSITION	Y_POSITION	FAILURE

Cuadro 3.10: Tabla *SECTOR_INFO*

Teniendo cada campo la funcionalidad expuesta a continuación:

- **ID**: Identificador único de entrada en la base de datos.
- **SECTOR_ID**: Identificador de sector. Se deberá utilizar la MAC[5] de la placa de control que lo gobierna.
- **FIBER_ID**: Identificador del actuador. Se deberá utilizar la dirección física del actuador.
- **X_POSITION**: Coordenada X.
- **Y_POSITION**: Coordenada Y.
- **FAILURE**: Bandera de estado del actuador. Si está a True, el campo será ignorado.

También hay que tener en cuenta que el campo *SECTOR_ID* y *FIBER_ID* forman un conjunto único, es decir, no puede haber dos entradas con un mismo identificador del actuador bajo el mismo identificador de sector.

La estructura anterior, en este proyecto, está implementada en una hoja de cálculo y, con el módulo escrito en Python, se adapta la misma para que pueda ser tratada por cada placa de control de manera eficiente.

[5] *Media Access Control*, Control de Acceso al Medio. Es un identificador de 48 bits que corresponde de forma única a un dispositivo de red.

El módulo de gestión de las coordenadas se encarga de separar los datos almacenados en la hoja de cálculo en diez ficheros CSV[6], uno por cada placa de control. Además tan solo se extraerán aquellos datos que tengan su campo *FAILURE* en *False*. De esta forma cada placa únicamente carga los datos que la pertenezcan y sólo aquellos que podrán ser enviados a los actuadores que no tengan fallos.

Continuado la con la primera parte del paquete software, se procede a recordar el algoritmo de configuración:

- Carga desde el PC de todas las coordenadas de la sección
- Selección del MUX
  - Selección de canal en el MUX
    - Envío de datos a cada uno de los 19 actuadores

El programa se encargará de recuperar del ordenador central las coordenadas correspondientes a la sección en la que está ubicada la placa de control. Esto se hace mediante la transferencia por SSH[7] del fichero CSV que contiene las coordenadas preparadas por el módulo de gestión de las coordenadas, explicado anteriormente.

Una vez que la placa de control reciba todos los datos, recordar que estos son siete parejas de coordenadas para cada actuador, procederá a configurar los multiplexores para establecer la comunicación con las celdas y, posteriormente, enviar las direcciones a cada actuador.

Cuando todos los actuadores han recibido sus siete parejas de coordenadas, el programa procederá a la lectura de los registros de los actuadores, pero esta parte se explica en el siguiente apartado.

Este proceso se repetirá cada treinta minutos.

[6] *Comma Separated Values*, es un tipo de documento que contiene las columnas separadas por comas.
[7] *Secure SHell*, protocolo que sirve para acceder a máquinas remotas y copiar ficheros a través de una red de forma segura utilizando cifrados.

3.2.3. Software - Comunicación por parte del esclavo

Además de enviar datos a los actuadores, también interesa leer datos de información de los mismos, tales como su estado, temperatura o la posición en la que se encuentran sus motores.

Puesto que el protocolo I^2C no permite utilizar interrupciones de forma sencilla, estos datos se obtendrán mediante lecturas de los actuadores. Para esto se enviará a cada uno de ellos un comando de consulta, que espera como respuesta la información solicitada.

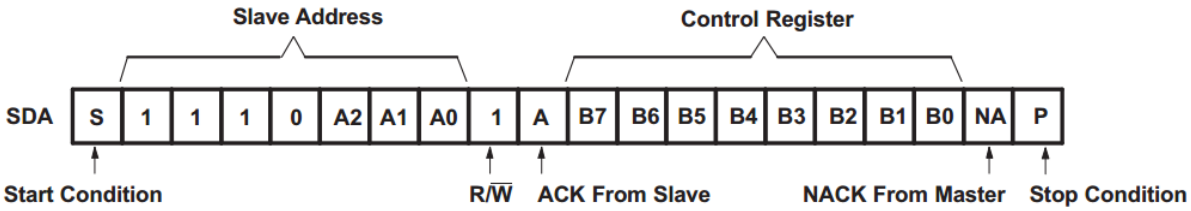

Figura 3.12: Trama de configuración para la recepción de datos

Cabe mencionar que en esta operación no hay ACK por parte de la placa de control al recibir los datos de los actuadores.

El algoritmo de consulta, para el esquema elegido, es el siguiente:

- Selección del MUX (9 bit)
    - Selección de canal en el MUX (9 bit)
        - Envío consulta de información al actuador (9 bit)
        - Recepción de información del actuador
- Envío de datos al ordenador central

Hay que destacar dos posibles casos:

- Información general: Estas consultas esperan una respuesta de 1 Byte
- Posición del actuador: Estas consultas esperan una respuesta de 2 Bytes

Se puede elaborar una tabla comparativa al igual que se ha hecho en el caso de envío de datos:

$\#Placas$	$T_{1\_actuador\_general}(s)$	$T_{1\_actuador\_posicion}(s)$	$T_{todos\_general}(s)$	$T_{todos\_posicion}(s)$
1	133×10^{-6}@$400KHz$	153×10^{-6}@$400KHz$	0.23 @ 400KHz	0.33 @ 400KHz
	53×10^{-6}@$1MHz$	61×10^{-6}@$1MHz$	0.09 @ 1MHz	0.13 @ 1MHz
2			0.115 @ 400KHz	0.165 @ 400KHz
			0.045 @ 1MHz	0.065 @ 1MHz
5	88×10^{-6}@$400KHz$	108×10^{-6}@$400KHz$	0.046 @ 400KHz	0.066 @ 400KHz
	35×10^{-6}@$1MHz$	43×10^{-6}@$1MHz$	0.018 @ 1MHz	0.026 @ 1MHz
10			0.023 @ 400KHz	0.033 @ 400KHz
			0.009 @ 1MHz	0.013 @ 1MHz

Cuadro 3.11: Tiempos estimados de lectura

Como era de esperar, se puede ver que el proceso de lectura de información de los actuadores es mucho más rápido que el de escritura de coordenadas. Esto es, en primer lugar, porque tan solo se recibe uno o dos bytes de información mientras que en el caso de escritura se envían 14 bytes de información.

También hay que destacar el hecho que, al igual que en la escritura de datos, al leer registros de los actuadores que se encuentran dentro de la misma celda, no hace falta reconfigurar el camino pues el MUX queda de forma transparente, por lo que el tiempo que se emplea en leer datos de todos los actuadores no corresponde a tomar el tiempo empleado en leer un único actuador y multiplicar por el número de actuadores.

3.3. Desarrollo

Para el desarrollo del sistema se han utilizado los programas *Altium Designer* y *NetBeans*, para la parte hardware y software respectivamente.

3.3.1. Hardware

Una vez decidido el diseño del sistema se ha procedido a escoger los componentes que van a formarlo.

- Como placa de control se ha tomado la tarjeta de desarrollo *Beagle Bone*[8], de BeagleBoard, que funciona sobre el sistema operativo UNIX. A fechas de comienzo de este proyecto, es una de las opciones que mejor relación tienen entre calidad y precio.

  Ofrece un procesador ARM Cortex A8 a 720 MHz, 256 MB de memoria RAM, Conexiones USB y Ethernet, además de un gran abanico de GPIOs.[9]

  En cuanto a la comunicación que interesa para este proyecto, esta placa es capaz de trabajar con velocidades de 400 Kbps en su bus I^2C.

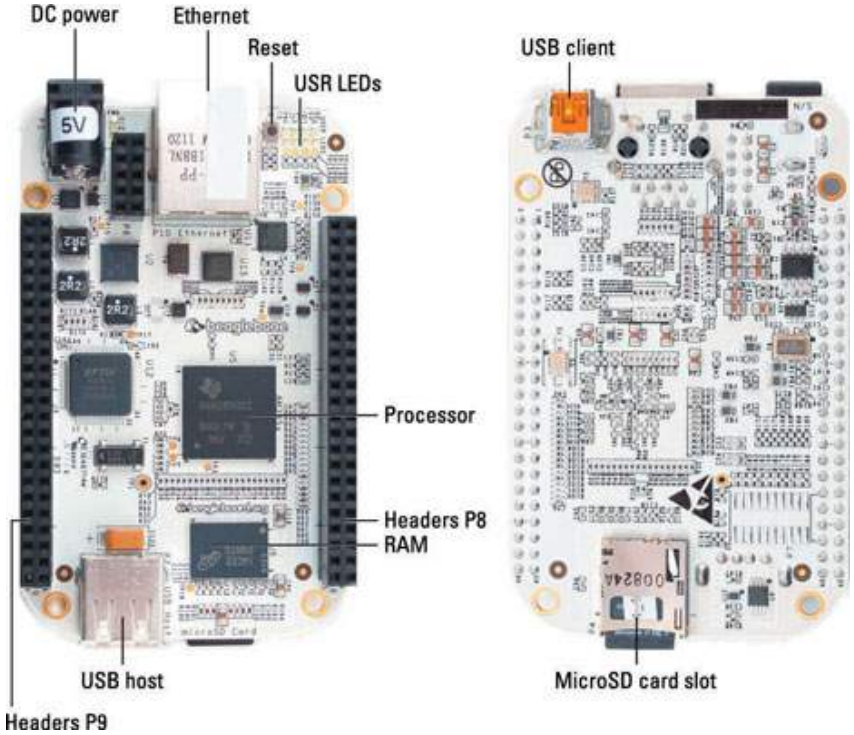

Figura 3.13: Beagle Bone

[8] Más información en: `http://goo.gl/ZiAsjB`

[9] *General Purpose Input Output*, es un pin genérico en un chip, cuyo comportamiento se puede programar por el usuario en tiempo de ejecución.

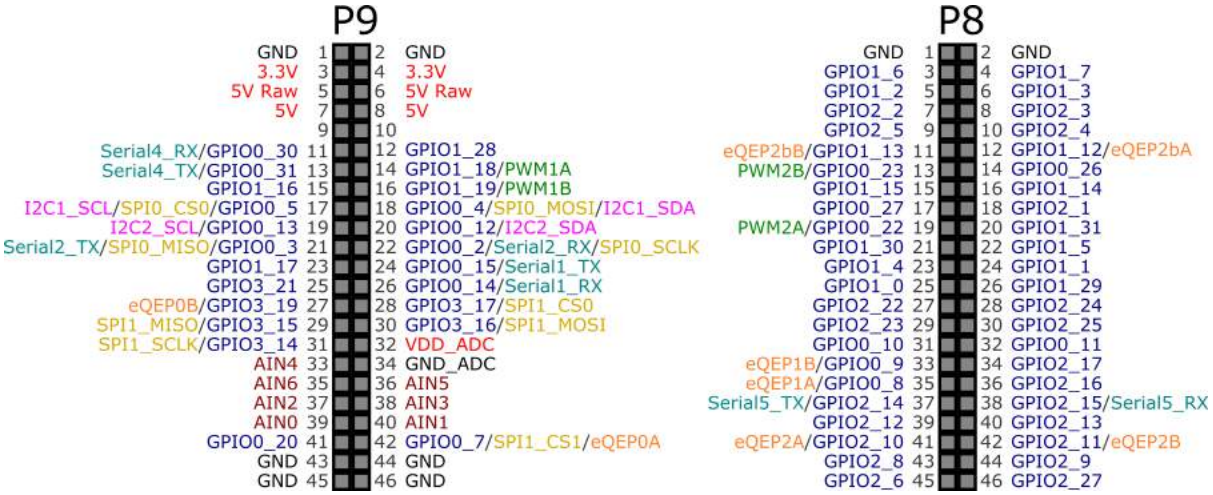

Figura 3.14: GPIO Pinout de Beagle Bone

- Puesto que en el análisis previo se han contemplado velocidades de 400 Kbps y 1 Mbps, se han buscado multiplexores que permitan realizar un circuito compatible con ambas opciones.

  Esto se ha conseguido con los chip:

  - *PCA9548A* de NXP[10], ofrece ocho salidas, una velocidad de 400 Kbps y tres bits de direccionamiento de chip.
  - *PCA9848* de NXP[11], ofrece ocho salidas, una velocidad de 1 Mbps y dos bits de direccionamiento de chip.

Si se observa el pinout de los multiplexores elegidos, puede verse enseguida que es posible diseñar un circuito que permita el uso de ambos modelos sin realizar cambios algunos en el circuito.

Para ello tan solo se debe fijar el pin 1 del PCA9548A, que corresponde al LSB[12] del direccionamiento, a la alimentación del circuito, tal como está conectado el correspondiente pin del PCA9848.

Realizando esta acción, tenemos las siguientes posibles direcciones:

- **PCA9548**: 1, 3, 5, 7
- **PCA9848**: 0, 1, 2, 3

[10] Datasheet: `http://goo.gl/p0g5o0`
[11] Datasheet: `http://goo.gl/DmNJ77`
[12] *Less Significant Bit*, el bit menos significativo.

Para poder utilizar un chip u otro, tan sólo se deberá configurar la rutina de acceso en el software, según el modelo seleccionado.

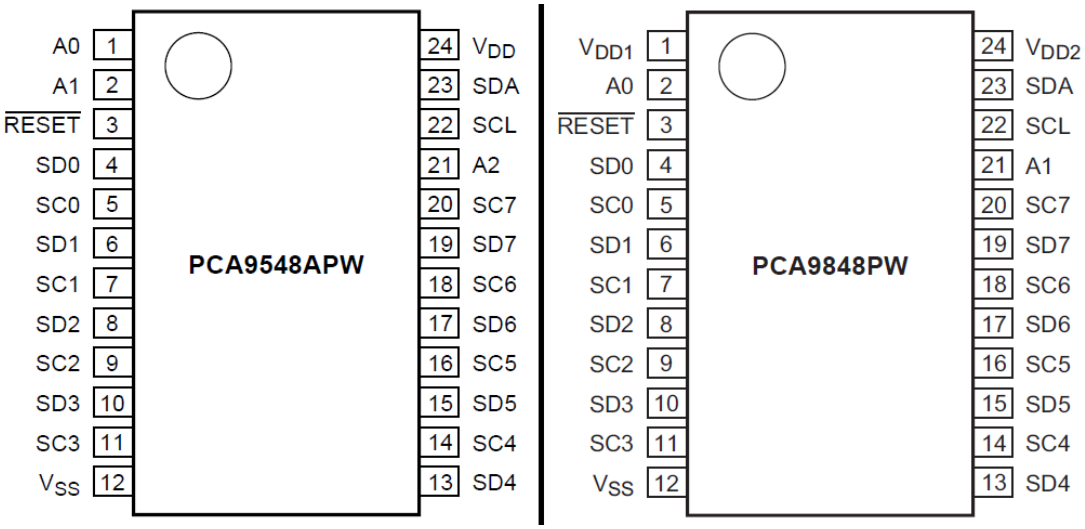

Figura 3.15: Pinout de los multiplexores elegidos

Tras presentar los componentes básicos del sistema, también hay que tener en cuenta ciertos aspectos de diseño de circuitos:

- Utilización de condensadores de desacoplo en los multiplexores.
- Utilización de resistencias pull-up en la línea de datos y del reloj.
- Tener el cuenta el efecto cross-talking a la hora del diseño.

Para verificar mínimamente la teoría presentada hasta ahora, se ha diseñado una placa de comunicación compuesta por dos MUX de 400 KHz, cada uno con dos canales habilitados. Será una placa intermedia entre la placa de control *BeagleBone* y la celda con los 19 esclavos. Con ella se pretenden comprobar los siguientes aspectos:

- Tiempos de comunicación y configuración de MUX
- Tiempos de selección de los canales dentro del MUX
- Tiempos de comunicación y configuración de un esclavo
- Tiempos de comunicación y configuración de una celda
- Tiempos de cambio de canal y de MUX

A continuación se presentan los esquemáticos de la placa fabricada.

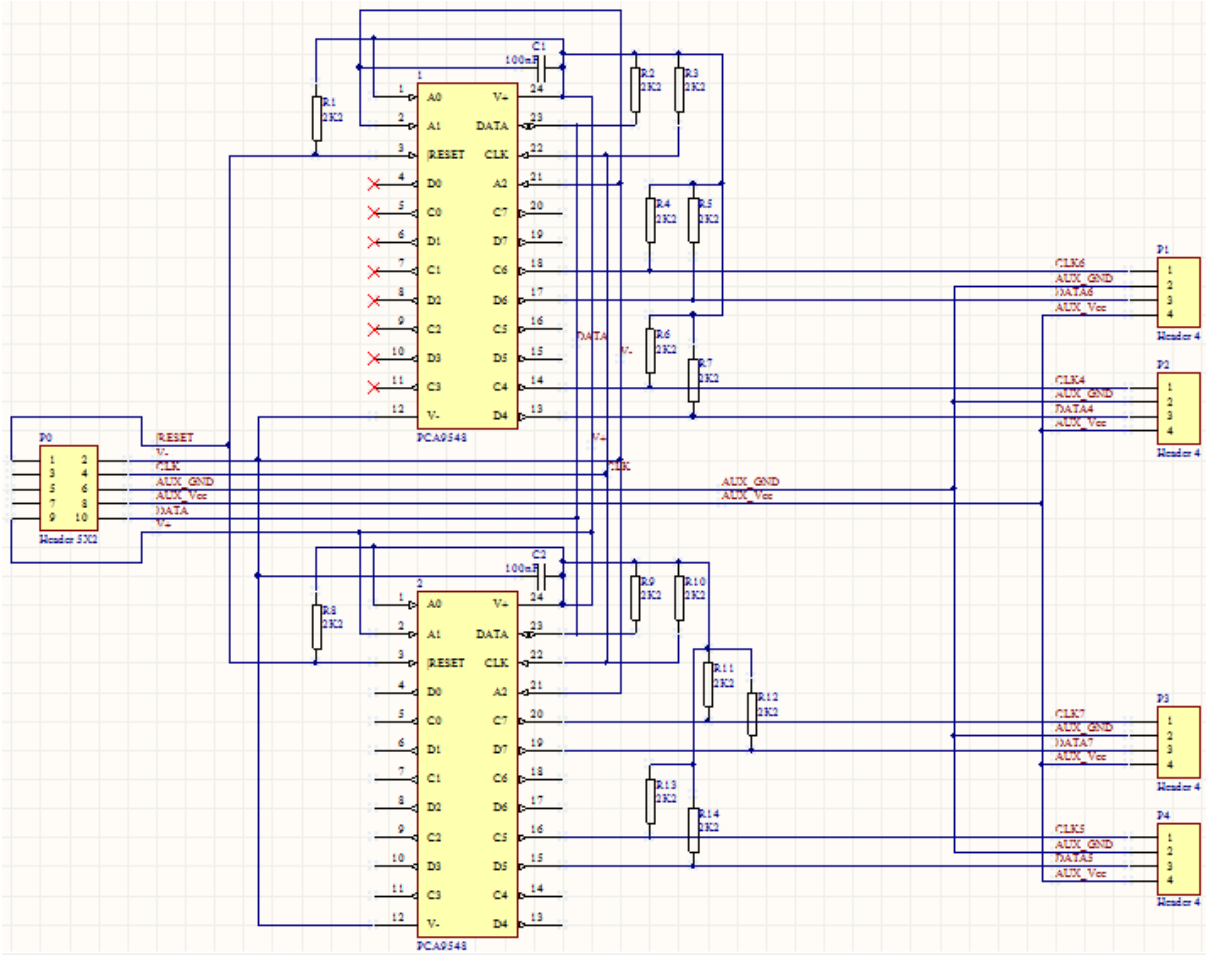

Figura 3.16: Esquemático de la placa de comunicación

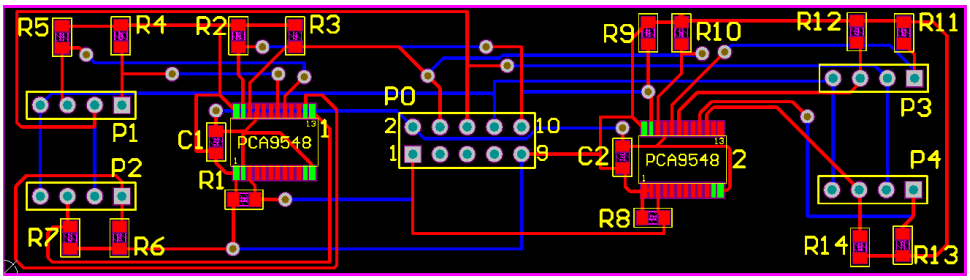

Figura 3.17: Diseño PCB de la placa de comunicación

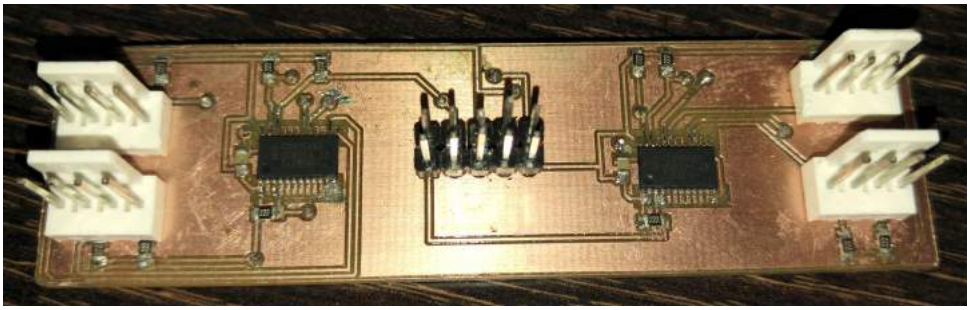

Figura 3.18: Placa de comunicación

Para realizar las pruebas anteriormente mencionadas sería necesario disponer del hardware utilizado en el proyecto final, es decir, los actuadores. Dado que no se dispone de los mismos, se utilizarán componentes que emularán ser los esclavos I^2C para permitir probar el diseño propuesto.

En concreto se ha utilizado una IMU[13] como esclavo independiente y una serie de memorias en una celda, que emularán a los actuadores.

La IMU utilizada es MinIMU-9[14] que contiene un giroscopio L3GD20H[15] y un acelerómetro LSM303D[16]. Puesto que es un dispositivo que contiene sensores de posición, servirá para poder comprobar la estabilidad del sistema ante una lectura constante de los registros.

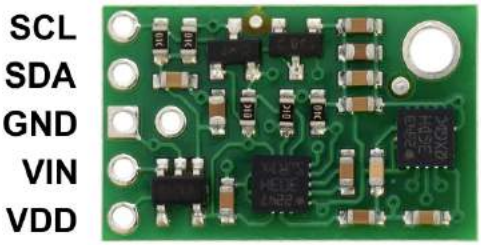

Figura 3.19: Pololu MinIMU-9 v3

[13]Del inglés *Inertial Measurement Unit*, Unidad de Medición Inercial. Es un dispositivo electrónico que mide la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos. IMU utilizada: https://goo.gl/BXPJnv

[14]Datasheet: https://goo.gl/zKC3Gh

[15]Datasheet: https://goo.gl/fWodr6

[16]Datasheet: https://goo.gl/zRqXcX

Las memorias que se han utilizado son 24C01[17] de Microchip. Con ellas se pretende emular la lectura y escritura en registros de los actuadores.

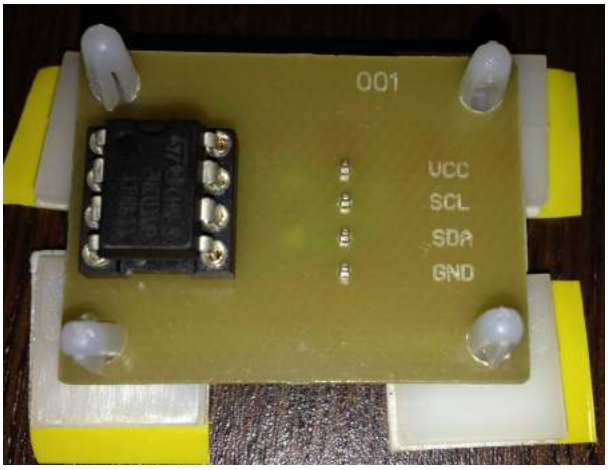

Figura 3.20: Memoria 24C01

Tal como se comentó anteriormente, las memorias se ubicarán en una celda, para lo que se ha fabricado una placa que cumpla la función de la misma.

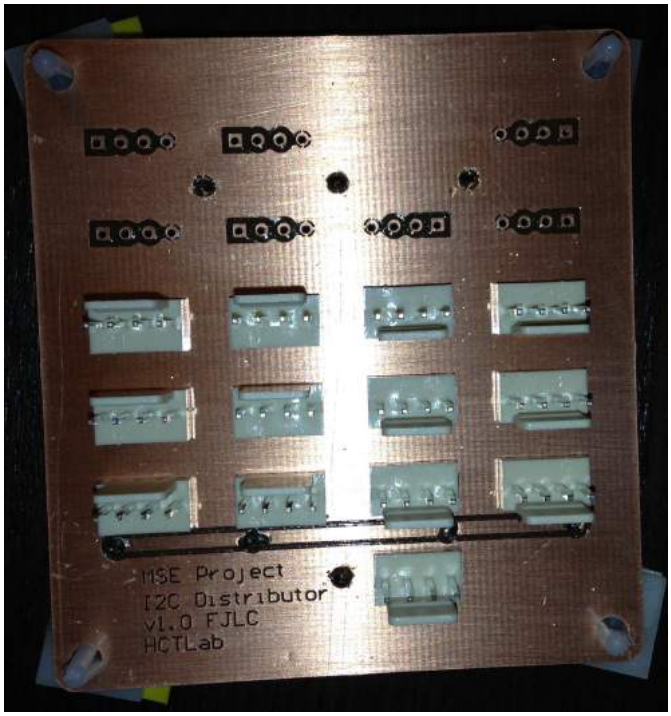

Figura 3.21: Adaptador de celda 1:19

[17] Datasheet: http://goo.gl/cif2MM

Una vez presentados todos los elementos que componen el sistema, se presenta el mismo en su totalidad.

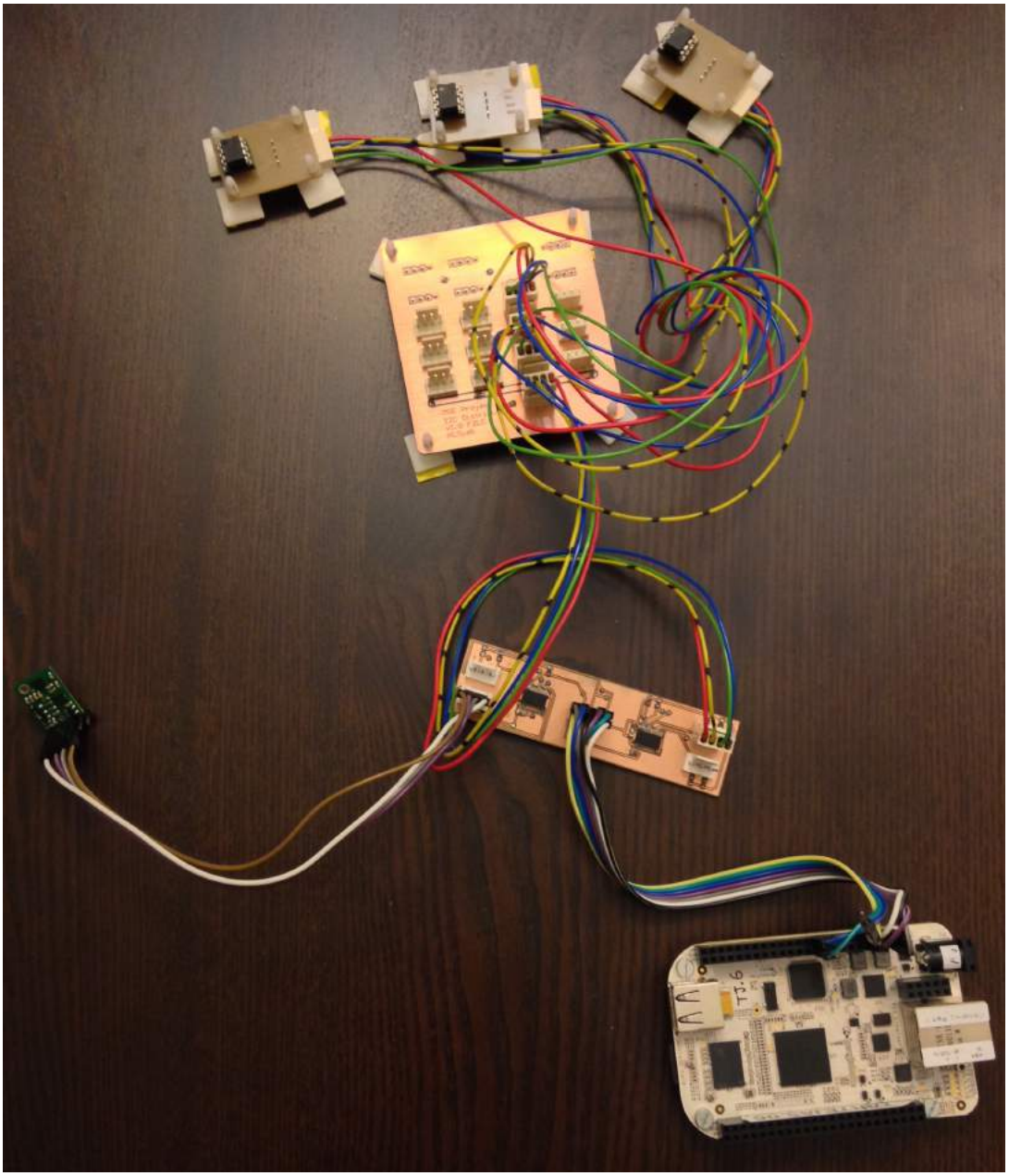

Figura 3.22: Sistema Completo

En la imagen puede observarse la placa de control *BeagleBone* conectada a la placa de comunicación. El IMU está conectado a un canal del primer MUX, mientras que el segundo MUX tiene conectada una celda, a la que se han conectado tres memorias que emulan a los actuadores.

3.3.2. Software

Tal como se comentó anteriormente, el paquete software se ha dividido en dos partes, una almacenada en las propias placas de control y otra en el ordenador central.

- Módulo Principal -

Este módulo se encarga de tomar las coordenadas desde el ordenador central y enviarlas a los actuadores correspondientes. Además, una vez completada la tarea anterior, se encarga de hacer un barrido por los actuadores asignados y leer su estado para después actualizarlo en el ordenador central.

La primera parte sigue el algoritmo presentado anteriormente. A modo de recordatorio:

- Carga desde el PC de todas las coordenadas de la sección
- Selección del MUX
    - Selección de canal en el MUX
        - Envío de datos a cada uno de los 19 actuadores

También se recuerda el algoritmo seguido para la segunda parte:

- Selección del MUX (9 bit)
    - Selección de canal en el MUX (9 bit)
        - Recepción de información de cada actuador en la celda

    Envío de datos al ordenador central

Para poder cumplir tal propósito, se estructura el código en siguientes ficheros:

- Archivos de configuración comunes, *headers*. Estos ficheros contienen las configuraciones y las declaraciones de funciones que se utilizarán por ambas partes del módulo.
    - **def_functions.h**: Contiene la definición de funciones utilizadas en el programa
    - **global_header.h**: Contiene la importación de todas las librerías utilizadas

- **HW_definition.h**: Este es el fichero de configuración que más atención debe llevarse, pues en él se configuran todas las rutinas que afectan al correcto funcionamiento del programa.

  - MAC : Variable global que contiene la MAC de la placa de control. Debe estar indicada con sus seis bloques hexadecimales separados por ":".

  - $I2C\_BUS$: Configura el bus I^2C que se utiliza en la placa de control. Debe estar indicada con un valor decimal entre 0 y 2.

  - $SECTION$: Representa la sección que gobierna la placa de control. Debe estar indicada con un valor decimal entre 1 y 10.

  - MUX_i : Representa la dirección física del MUX, siendo i variable entre 1 y 4. Debe estar indicada con un valor hexadecimal y ser acorde al modelo del chip escogido.

  - CH_i : Siendo i variable entre 0 y 7, contiene el valor con el que se escoge un canal u otro dentro del MUX. Debe estar indicada con un valor hexadecimal y seguir una codificación *one-hot*[18]

  - MUX_SIZE : Variable que define el número de los MUX utilizados en el circuito. En la versión de prueba este valor equivale a 2, mientras que en la versión final es 4.

  - $MUX\_CH\__i$: Siendo i X o Y, estas variables definen la matriz de canales disponibles en el circuito. En la versión de prueba esta matriz es de 2x2, pues hay dos MUX y cada uno de ellos ofrece dos salidas. En la versión final, la dimensión es 4x7.

  - $COORDINATE\_{i}$: Siendo i X o Y, estas variables definen la matriz que alberga las coordenadas. Recordando que en cada celda hay 19 actuadores, en cada sección hay 27 celdas y que cada actuador recibe 7 parejas de coordenadas, se obtiene que en total hay 3591 entradas. Teniendo en cuenta que para cada entrada se necesitan dos coordenadas y la dirección física del actuador al que corresponden, se tiene que la matriz de la versión final tiene una dimensión de 3591x3.

  - $CELL\_SIZE$: Indica la cantidad de actuadores que hay presentes en una celda. En el presente proyecto, este valor se considera 19.

  - $TIME\_SNOOZE$: Define el tiempo en minutos que hay entre cada ejecución del código. En pruebas este valor se establece de 1 minuto, mientras que en la versión final es de 30 minutos.

[18] *one-hot*: es un tipo de codificación secuencial en el que, en toda la cadena de bits, sólo hay un 1. La posición del 1 indica el número de la secuencia.

- **ssh_definition.h**: Contiene los parámetros de conexión SSH del ordenador central. Dos campos merecen especial atención:
  - *REMOTE_BASE_DIR* : Indica el directorio donde la placa de control espera encontrar el fichero de las coordenadas.
  - *LOCAL_BASE_DIR* : Indica el directorio en el que la placa de control almacenará el fichero de las coordenadas.

- Los archivos de transferencia de datos **hacia** los actuadores, que contienen todo el código necesario para recuperar las coordenadas desde el ordenador central y enviarlas a los mismos, son:
  - **tcp_ip_data_read.c**: Utilizando el protocolo SSH, la tarea de esta función es conectarse al ordenador central para obtener el fichero que contiene las coordenadas correspondientes a la celda asignada a la placa de control. Una vez que dicho fichero se encuentra en la placa de control, se trata su contenido para obtener la matriz de coordenadas que utilizará *mux_write.c*.
  - **mux_write.c**: Haciendo uso de *tcp_ip_data_read.c*, recibe la matriz con las coordenadas que se necesitan enviar a los actuadores de la sección. Una vez que se tienen estos datos, la función principal de este bloque es recorrer cada uno de los MUX seleccionando cada una de sus salidas y, mediante *positioner_write.c*, enviar las coordenadas a cada uno de los actuadores.
  - **positioner_write.c**: Esta función se encarga de preparar las siete parejas de coordenadas correspondientes al MUX que recibe como parámetro y enviarlas al mismo mediante I^2C.

- Los archivos de transferencia de datos **desde** los actuadores, que contienen todo el código necesario para obtener lecturas de los actuadores y enviar los datos relevantes al ordenador central, son:
  - **mux_read.c**: Esta función se encarga de recorrer uno por uno todos los actuadores de la sección y realizar lecturas de sus registros para obtener información sobre el estado de éstos. Este proyecto se ha centrado sobre el registro que contiene información del estado de fallo del actuador. De este modo, si se detecta que un actuador está roto, se procederá a informar de ello al ordenador central para que actualice el campo *FAILURE* de los actuadores correspondientes a *True*. Con esto se consigue que, en la siguiente lectura de coordenadas, los actuadores rotos se ignoren.
  - **tcp_ip_data_write.c**: Una vez que se identifican los actuadores en mal estado, esta función se encarga de comunicarlos al ordenador central para las actualizaciones pertinentes. Al igual que en *tcp_ip_data_read.c*, se utiliza el protocolo SSH.

- Otros archivos de interés son:

  - **get_mac.c**: Función que se encarga de obtener la MAC de la placa de control.
  - **str2hex.c**: Función que se encarga de traducir valores hexadecimales escritos con formato texto a formato hexadecimal.
  - **main.c**: Función principal del módulo, se encarga de repetir las tareas de escritura y lectura de datos con el intervalo temporal establecido en *TIME_SNOOZE*.

- Módulo de Gestión de Coordenadas -

Este módulo, escrito en un único fichero, se encarga de tomar la hoja de cálculo y, mediante Python y la librería *openpyxl*, adaptarla a los ficheros CSV correspondientes a cada placa de control.

La adaptación se basa en los campos *SECTOR_ID* y *FAILURE*, siendo el primer campo el indicativo a qué sector pertenece la entrada de la tabla y el segundo campo, el indicativo a si el dato tiene que ser tomado en cuenta o ignorado.

4

Experimentos Realizados y Resultados

El sabio puede sentarse en un hormiguero, pero sólo el necio se queda sentado en él.

Proverbio chino.

En el presente capítulo se recogen los experimentos Realizados con el sistema desarrollado.

Tal como se expuso en el anterior capítulo, se pretende comprobar la viabilidad del sistema teniendo en cuenta:

- Tiempos de comunicación y configuración de MUX
- Tiempos de selección de los canales dentro del MUX
- Tiempos de comunicación y configuración de un esclavo
- Tiempos de comunicación y configuración de una celda
- Tiempos de cambio de canal y de MUX

Para ello se van a realizar las siguientes pruebas:

- 1 - Escaneo del bus I^2C en búsqueda de esclavos
- 2 - Configuración de un canal en un MUX
- 3 - Lectura de datos de un esclavo conectado al MUX
- 4 - Cambio de MUX y canal
- 5 - Escritura y Lectura de datos en un esclavo conectado a una celda
- 6 - Escritura de datos en una celda completa

Antes de proceder con las pruebas hay que mencionar que no se ha logrado hacer funcionar el BUS I^2C de la placa de desarrollo BeagleBone a la frecuencia esperada de 400KHz, por lo que la tabla de los resultados teóricos correspondiente a utilizar una placa por sección se ha adaptado a la frecuencia de 100KHz, que es la que se ha logrado conseguir en dicha placa.

Tarea	**bits**	**Velocidad**	**Tiempo(s)**	**Comentarios**
Envío de datos PC -> Placa	57456	1Gbps	57×10^{-6}	Se reciben los datos del PC
Configuración MUX	36	100Kbps	360×10^{-6}	Tiempo de configuración de todos los MUX
Configuración Canal	243	100Kbps	2430×10^{-6}	Tiempo de configuración de las salidas de MUX
Envío de datos Placa -> Actuadores	64638	100Kbps	0,65	Tiempo de envío de datos a los actuadores + ACK

Cuadro 4.1: Tiempos estimados en cada tarea

Se utilizará el BUS 1 de la placa de control *BeagleBone Black*, funcionando a 100KHz, para realizar todas las pruebas.

Para analizar las tramas de comunicación se ha utilizado el osciloscopio *InfiniiVision 3000T X-Series*[1].

[1] Datasheet: http://goo.gl/snIOU8

4.1. Escaneo del BUS I^2C en búsqueda de esclavos

En primer lugar se realizará un escaneo del BUS sin conectar la placa de comunicación y, posteriormente, se realizará otro escaneo con la placa de comunicación conectada. Se recuerda que las direcciones que se han configurado para los MUX en esta placa han sido 001 y 011, por lo que se espera que los dispositivos detectados en el BUS sean 0x71 y 0x73.

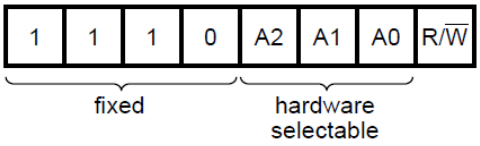

Figura 4.1: Dirección de MUX PCA9548

- Se realiza el escaneo del BUS vacío

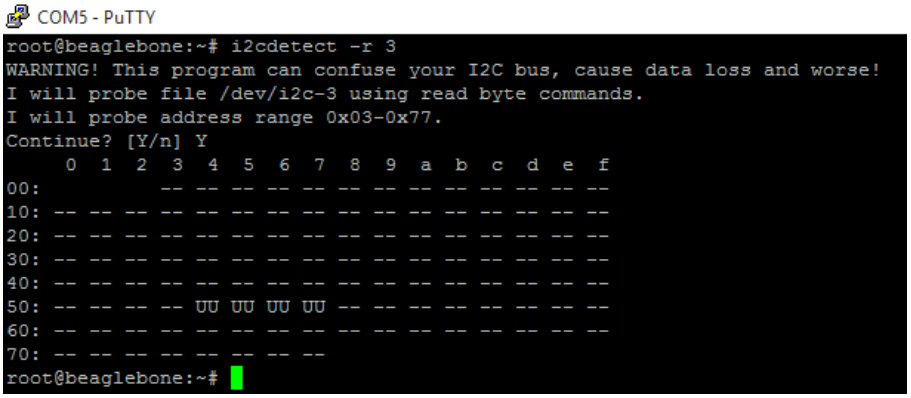

Figura 4.2: Escaneo del BUS vacío

- Se procede a conectar la placa de comunicación y a realizar un nuevo escaneo

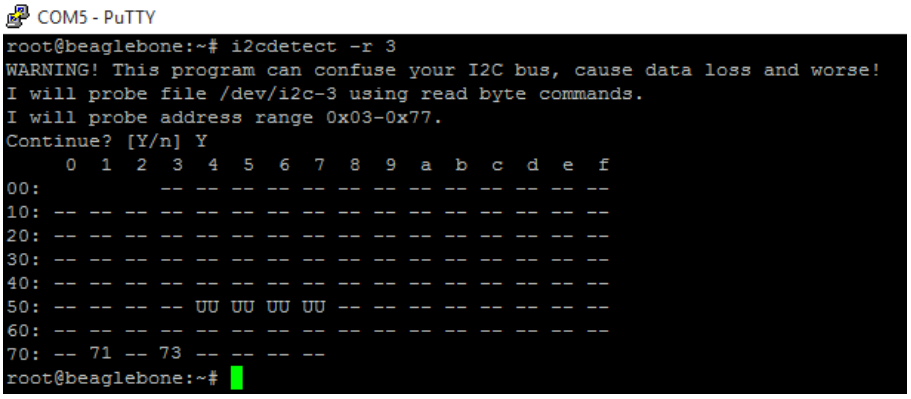

Figura 4.3: Escaneo del BUS con la placa de comunicación conectada

Puede verse que el resultado obtenido es el esperado.

Analizando la trama con el osciloscopio se puede ver que esta acción se ha realizado en un tiempo de 50.756ms.

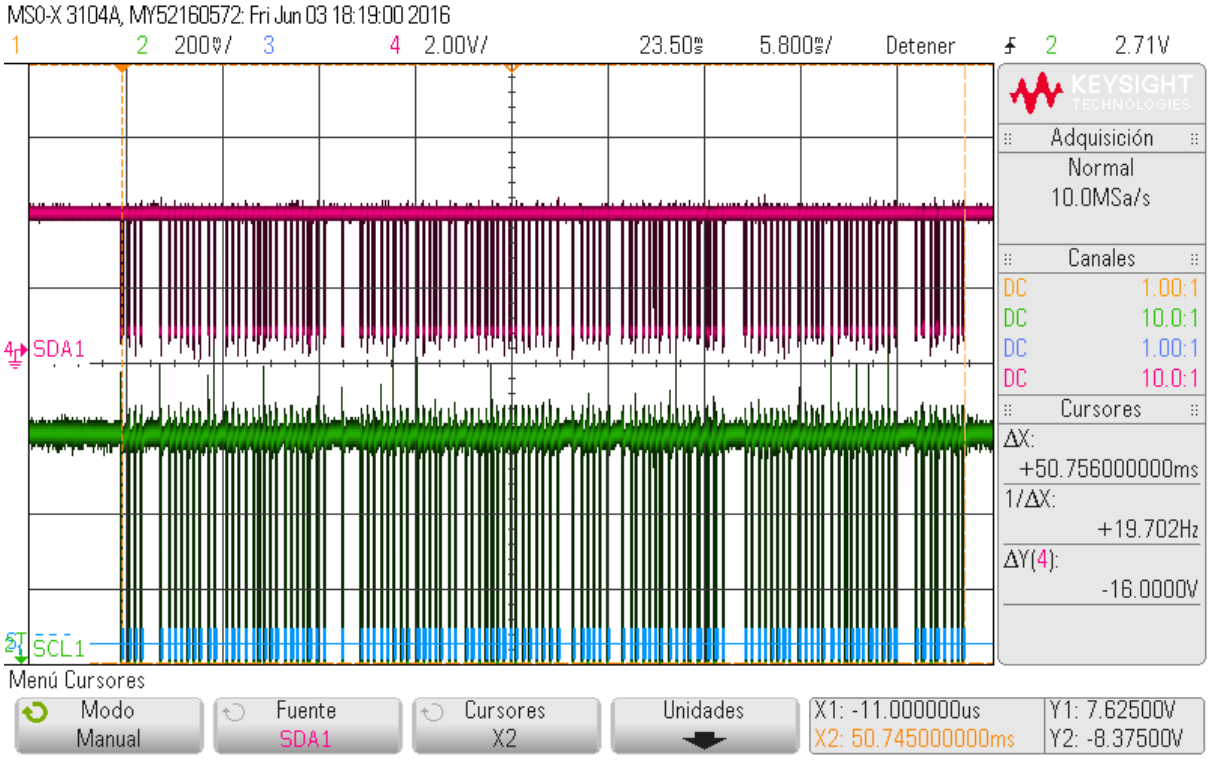

Figura 4.4: Análisis de trama con osciloscopio

Observando en detalle la captura, cabe destacar que no se ha enviado toda la información de manera uniforme en el tiempo, sino que existen pausas de mayor duración en ciertos puntos del envío de datos. Esto puede deberse a operaciones internas de la propia placa de control *BeagleBone* durante las que la comunicación del BUS I^2C queda pausada.

En este primer paso se ha comprobado una correcta detección de la placa de comunicación por parte de la placa de control, por tanto se procede a la siguiente prueba.

4.2. Configuración de un canal en un MUX

Una vez comprobada la detección de la placa de comunicación, se procede a habilitar el canal 4 del MUX 0x71. Para habilitar un canal se debe colocar un 1 en el bit correspondiente al mismo, por lo que el canal 4 corresponde al byte 0x10.

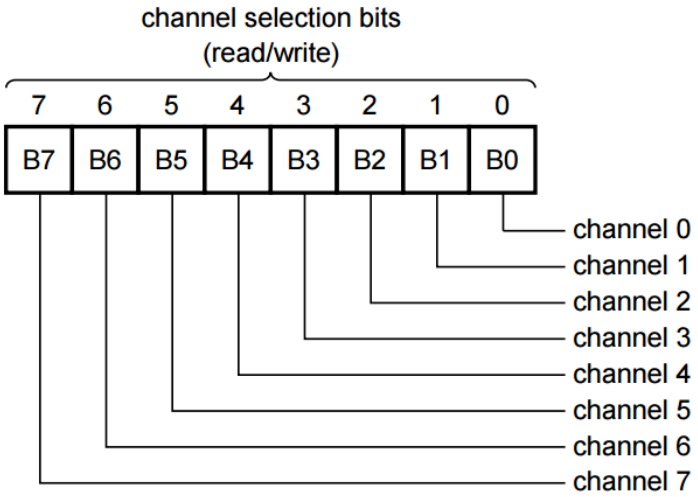

Figura 4.5: Bits de selección de canal

Este canal tiene conectado el IMU mencionado en el capítulo anterior, que a su vez tiene el acelerómetro en la dirección 0x1d y el giroscópio en la dirección 0x6b, direcciones que se esperan ver al realizar un nuevo escaneo del BUS.

COM5 - PuTTY

```
root@beaglebone:~# i2cset -y 3 0x71 0x10
root@beaglebone:~# i2cdetect -r 3
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-3 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] Y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- 1d -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- 6b -- -- -- --
70: -- 71 -- 73 -- -- -- --
root@beaglebone:~#
```

Figura 4.6: Configuración de la salida del MUX y posterior escaneo

Como puede observarse, los dispositivos esperados aparecen ahora en el BUS, por lo que el canal se ha configurado correctamente.

Analizando la trama con el osciloscopio se observa que el tiempo total empleado en el proceso es de 201.5us. Al tratarse de una operación de escritura, también se puede ver un ACK confirmando el dato, 0x10.

Un dato a destacar es que se han procesado 18 bits, 9 de los cuales corresponden al comando de escritura más su ACK y otros 9 al byte enviado más su ACK. Es importante tener esto en cuenta ya que, a pesar de que el proceso completo se ha llevado a cabo en 201.5us, la mitad de este tiempo ha sido invertido en enviar el propio comando de escritura. Por tanto el byte que se ha enviado, más el ACK correspondiente, ha necesitado 100.75us para llegar a su destino.

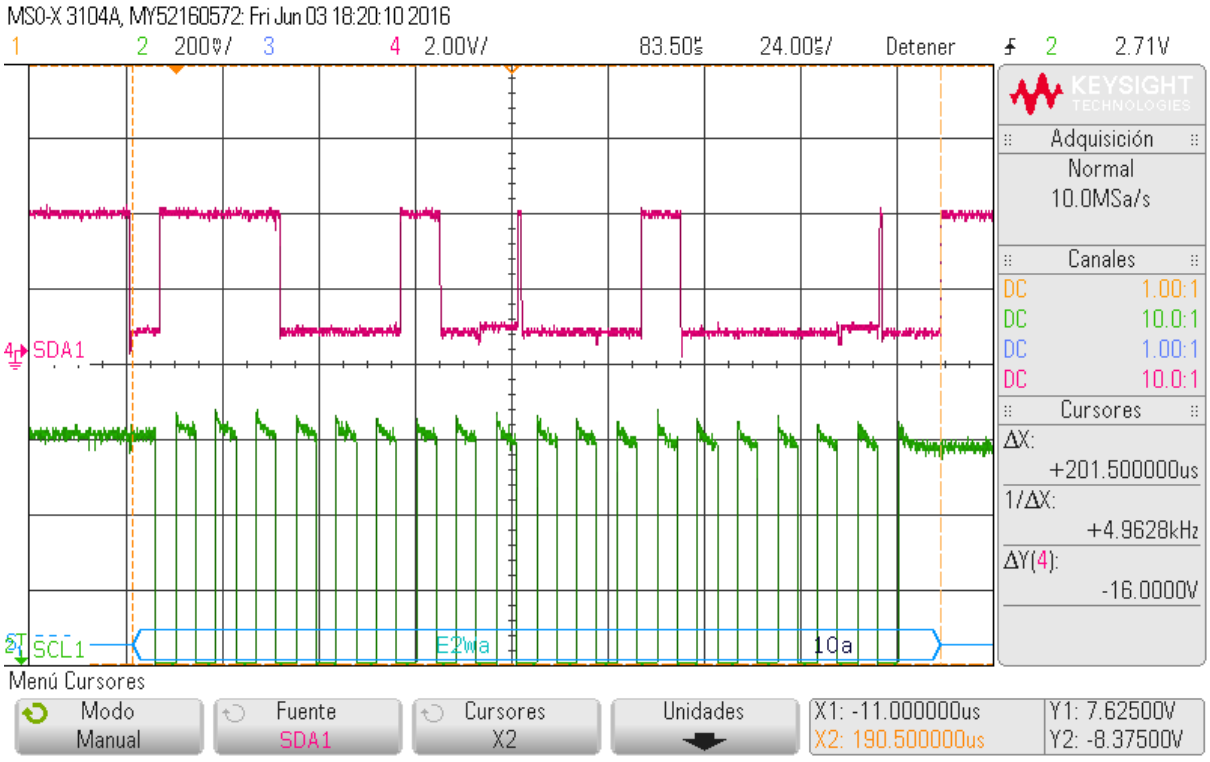

Figura 4.7: Tiempo de configuración del canal 4 en MUX 0x71

Una vez establecido el canal en el MUX, este queda de forma transparente para la aplicación y se puede interactuar diréctamente con los dispositivos del IMU, proceso que se realizará en el siguiente paso.

4.3. Lectura de datos de un esclavo conectado al MUX

En este apartado se procederá a interactuar con los dispositivos del IMU, en concreto se leerán algunos de sus registros. Para asegurarse de que el valor leído es el correcto, se leerán los registros prefijados de cada dispositivo.

En primer lugar se leerá el registro de identificación del acelerómetro. Para eso hay que consultar el registro 0x0f del dispositivo 0x1d y, como respuesta, se espera el valor 01001001 (0x49).

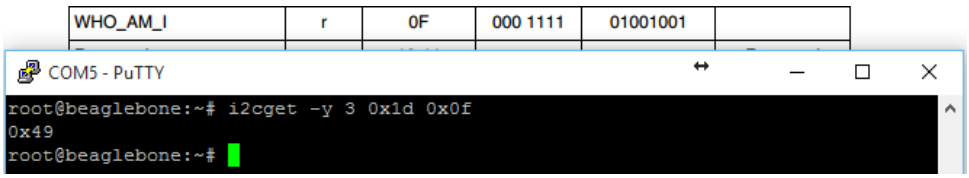

Figura 4.8: Consulta del registro de identificación del acelerómetro

Analizando la trama con el osciloscopio, podemos diferenciar dos partes:

- Se establece un comando de escritura, con el correspondiente ACK
- Solicitud de lectura del registro 0x0f, con su correspondiente ACK
- Se establece una condición de *Start* de lectura, con el correspondiente ACK
- Por útlimo se recibe la respuesta con el dato 0x49, sin ACK

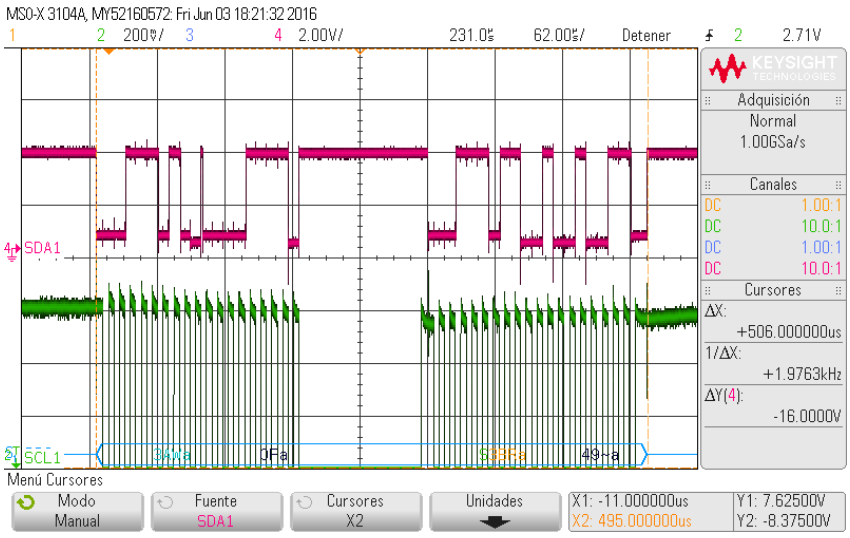

Figura 4.9: Tiempo de consulta del registro de identificación

El tiempo total de la operación, durante la que se ha enviado un byte y se ha recibido otro, ha sido de 506us. Este valor es coherente con el obtenido en el punto anterior, en

el que se han empleado 201.5us para la trasmisión de un byte, si se tiene en cuenta la parada que de 105us que hay entre la recepción de la pecitión del dato y el envío del mismo.

A continuación se realizará la misma prueba con el segundo dispositivo del IMU, el giroscopio. Para ello se consultará el mismo registro del dispositivo 0x6b y, como respuesta, se espera el valor 11010111 (0xd7).

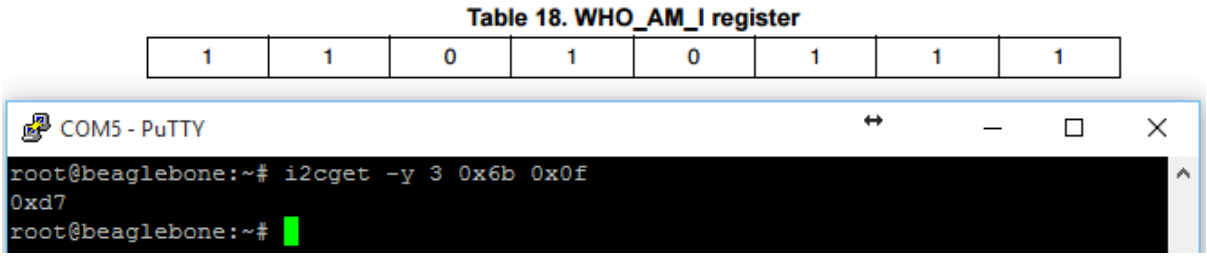

Figura 4.10: Consulta del registro de identificación del giroscópio

Una vez más se comprueba que se obtienen los resultados esperados y, contrastando con la información obtenida con el osciloscopio, se observa que los tiempos son coherentes con la operación.

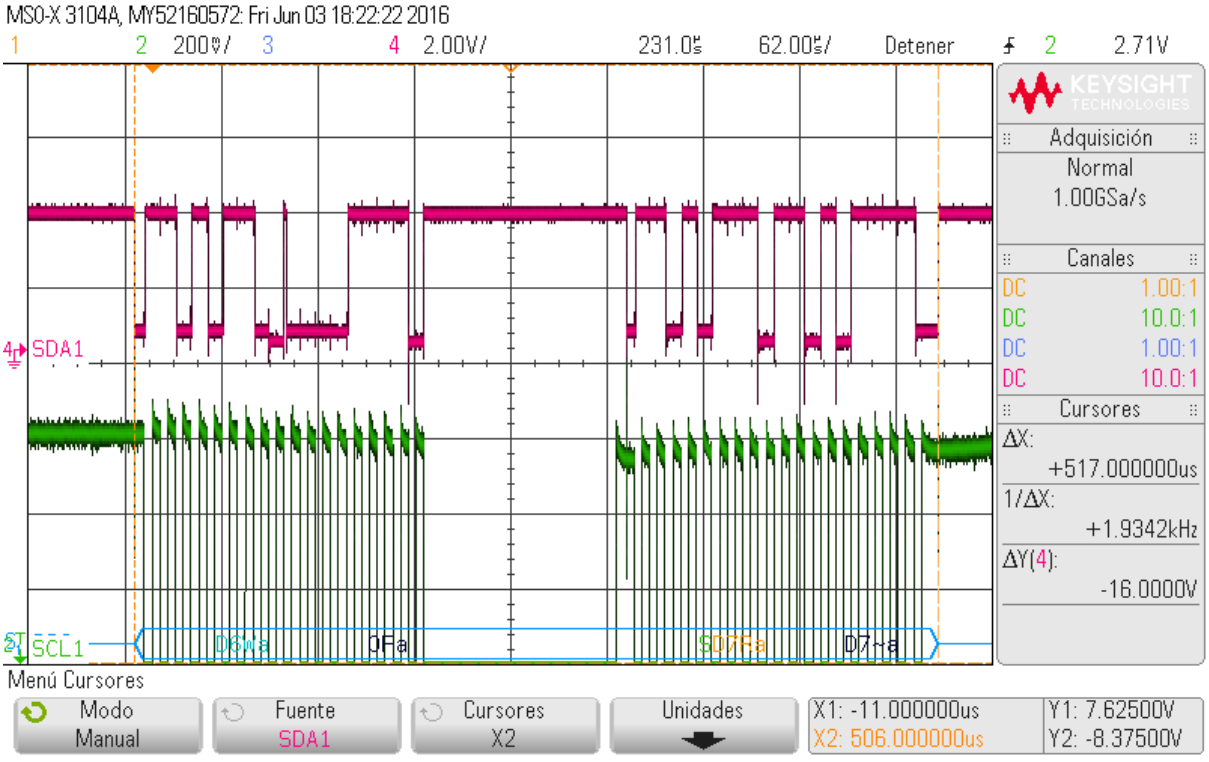

Figura 4.11: Tiempo de consulta del resitro de identificación

Realizadas las pruebas con un único esclavo conectado al MUX, se proceden a realizar las pruebas con un conjunto de esclavos conectados a una celda.

4.4. Cambio de MUX y canal

La celda está conectada al canal 7 del MUX 0x73 por lo que, basándose en la *figura 4.5*, se debe escribir el byte 0x80 en dicho MUX.

Las direcciones utilizadas en las memorias 24C01 han sido 000, 001 y 010. Teniendo en cuenta la hoja de características, una vez configurado el canal, se espera encontrar tres dispositivos nuevos en el BUS, en las direcciones 0x50, 0x51 y 0x52.

Table 3. Device select code

	Device Type Identifier(1)				Chip Enable(2),(3)			RW
	b7	b6	b5	b4	b3	b2	b1	b0
M24C01 Select Code	1	0	1	0	E2	E1	E0	RW
M24C02 Select Code	1	0	1	0	E2	E1	E0	RW
M24C04 Select Code	1	0	1	0	E2	E1	A8	RW
M24C08 Select Code	1	0	1	0	E2	A9	A8	RW
M24C16 Select Code	1	0	1	0	A10	A9	A8	RW

1. The most significant bit, b7, is sent first.
2. E0, E1 and E2 are compared against the respective external pins on the memory device.
3. A10, A9 and A8 represent most significant bits of the address.

Figura 4.12: Código de selección de chip en memoria 24C01

```
root@beaglebone:~# i2cset -y 3 0x73 0x80
root@beaglebone:~# i2cdetect -r 3
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-3 using read byte commands.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 51 52 -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- 71 -- 73 -- -- -- --
root@beaglebone:~#
```

Figura 4.13: Configuración de canal y detección de memorias

Tras la reconfiguración del MUX se han encontrado los tres dispositivos esperados y, al igual que en el caso de un esclavo conectado diréctamente al canal del MUX 0x71, el MUX 0x73 queda de forma transparente para la aplicación.

Al tratarse de una operación equivalente a la realizada en la prueba **2**, los tiempos son iguales, pues no dependen del dispositivo sino del BUS.

Una vez establecida la comunicación con los componentes de la celda, se procede a interactuar con los mismos.

4.5. Escritura y Lectura de datos en un esclavo conectado a una celda

Esta prueba consiste en escribir un byte en la memoria y, posteriormente, leer el registro para comprobar que el dato que se ha guardado es el que realmente se ha enviado.

En esta prueba, la escritura en memoria emula la escritura de una pareja de coordenadas en el actuador, mientras que la lectura de la memoria emula la lectura de un registro de estado del actuador.

Concretamente se procederá a escribir el valor 0xaa en el registro 0x10 de la memoria 0x50 y el valor 0x1a en el registro 0x11 de la memoria 0x51. Una vez escritos los valores en los registros se procederá a leerlos para verificar su validez.

```
root@beaglebone:~# i2cset -y 3 0x50 0x10 0xaa
root@beaglebone:~# i2cget -y 3 0x50 0x10
0xaa
root@beaglebone:~#
```

Figura 4.14: Escritura y lectura en memoria 0x50

Tras comprobar que el valor leído del registro corresponde con el que se le envió, se procede a analizar la trama de escritura y, posteriormente, la de lectura.

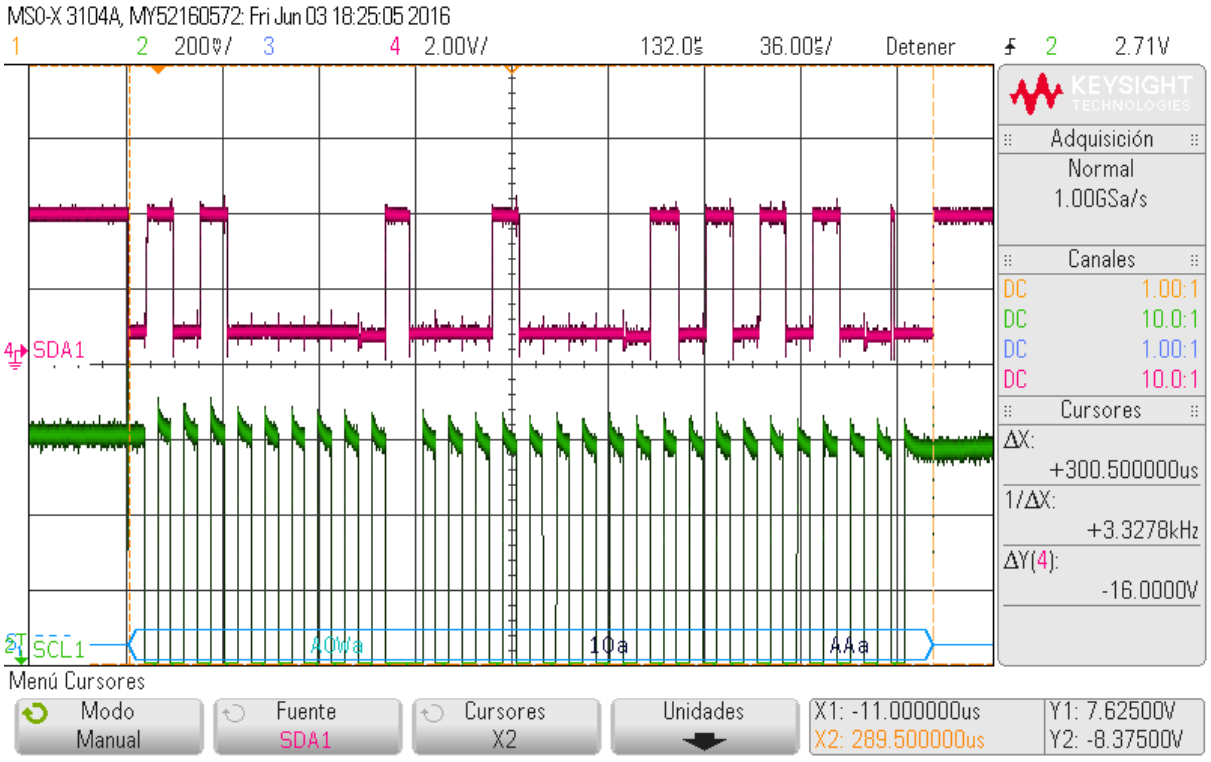

Figura 4.15: Trama de escritura en memoria 0x50

En la trama de escritura se puede observar una primera parte que corresponde al propio comando de escritura, con el correspondiente ACK, una segunda parte que contiene

la dirección del registro en el que se va a escribir, con el correspondiente ACK, y por último una tercera parte con el propio valor que se almacena en el registro, también con el correspondiente ACK. El tiempo total empleado en esta operación ha sido de 300.5us, un valor razonable teniendo en cuenta que la escritura de un byte necesita aproximádamente 200us.

Si se observa en detalle la captura realizada por el osciloscópio, puede verse que existe una pequeña parada entre la recepción del comando de escritura por parte de la memoria y el envío de datos a la misma. Esta parada corresponde a 9us, que representa el 3 % del tiempo total empleado.

Por otra parte cabe volver a destacar que además de los bytes enviados a la memoria también se envía el propio comando de escritura, que corresponde a un byte con su correspondiente ACK. Este comando requiere 99us para transmitirse. Esto representa el 32.95 % del tiempo total empleado.

Con estos datos se tiene que los dos bytes enviados han necesitado 192.5us para llegar a la memoria, que es el 64.05 % del tiempo total empleado en la acción realizada.

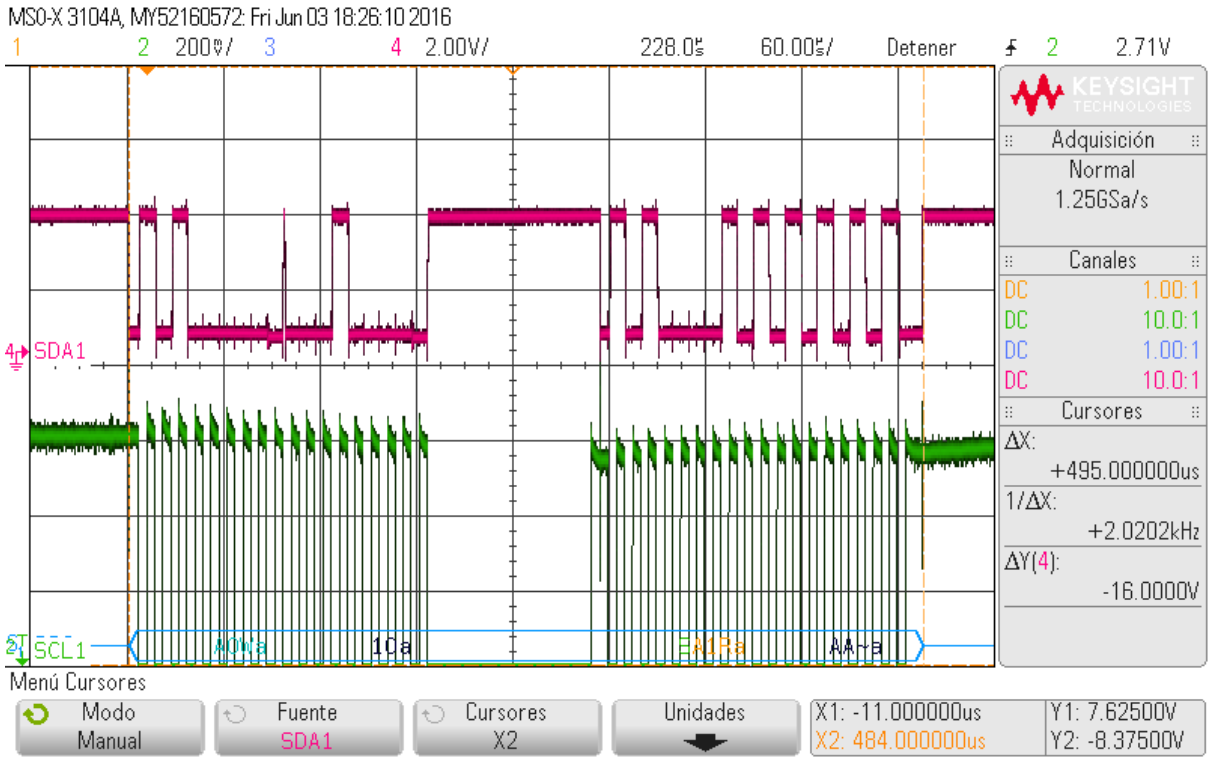

Figura 4.16: Trama de lectura en memoria 0x50

En la trama de lectura se puede diferenciar una parte que corresponde a la consulta del registro de interés, 0x10, con el ACK correspondiente, otra que corresponde al inicio de envío de datos por parte del esclavo, con el correspondiente ACK, y por último el propio dato, 0xaa, sin ACK. Esta operación se ha llevado a cabo en un tiempo de 495us. Al igual que en la prueba **3**, es un valor de esperar pues se trabaja con dos bytes.

Realizada la prueba de escritura de un byte en la memoria, se procede a realizar una escritura de todas las coordenadas sobre la misma. Recordar que cada actuador requiere dos coordenadas y, al utilizar el algoritmo de siete pasos, se envían siete parejas de coordenadas, por lo que en total se envían 14 bytes por actuador.

Para la realización de esta prueba se utiliza el software desarrollado, preparando las rutinas para adaptarlo a las memorias, que emulan ser los actuadores.

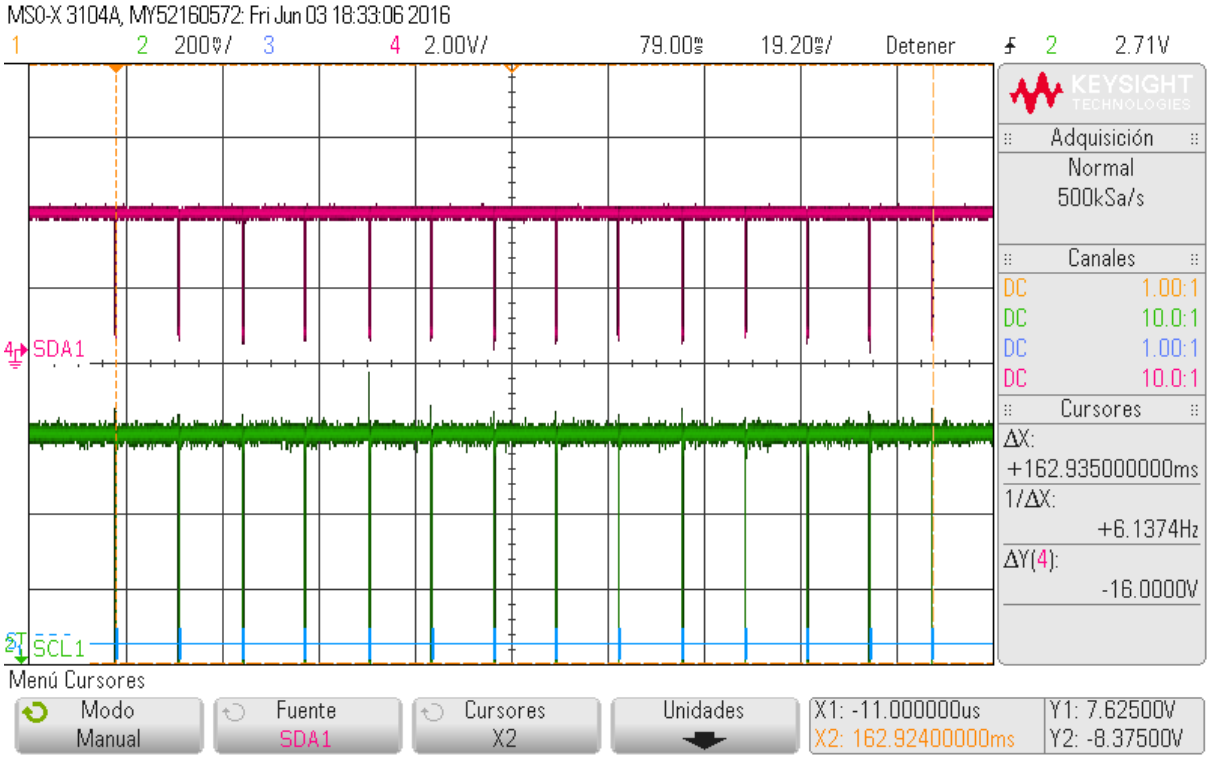

Figura 4.17: Escritura de 14 coordenadas en memoria

Se puede ver que el tiempo total en escribir los 14 bytes ha sido de 162.935ms, sin embargo no todo el tiempo se ha empleado en la transmisión de los datos.

Si se observa la gráfica en detalle, puede verse que cada byte no se envía a continuación del anterior, sino que hay un tiempo de espera, debido a operaciones internas de la placa de control y los tiempos de guardado del dato en memoria.

Este tiempo es de 12.04ms y, en su totalidad, representa el 96.06 % del tiempo total invertido en escribir los 14 bytes en memoria. Si se resta este tiempo se obtiene que el tiempo invertido únicamente en la transmisión de los datos ha sido de 6.415ms.

Si además se tiene en cuenta que los datos que interesan requieren el 64.05 % de ese tiempo, se tiene que el tiempo de transmisión de las coordenadas ha sido de 4.109ms.

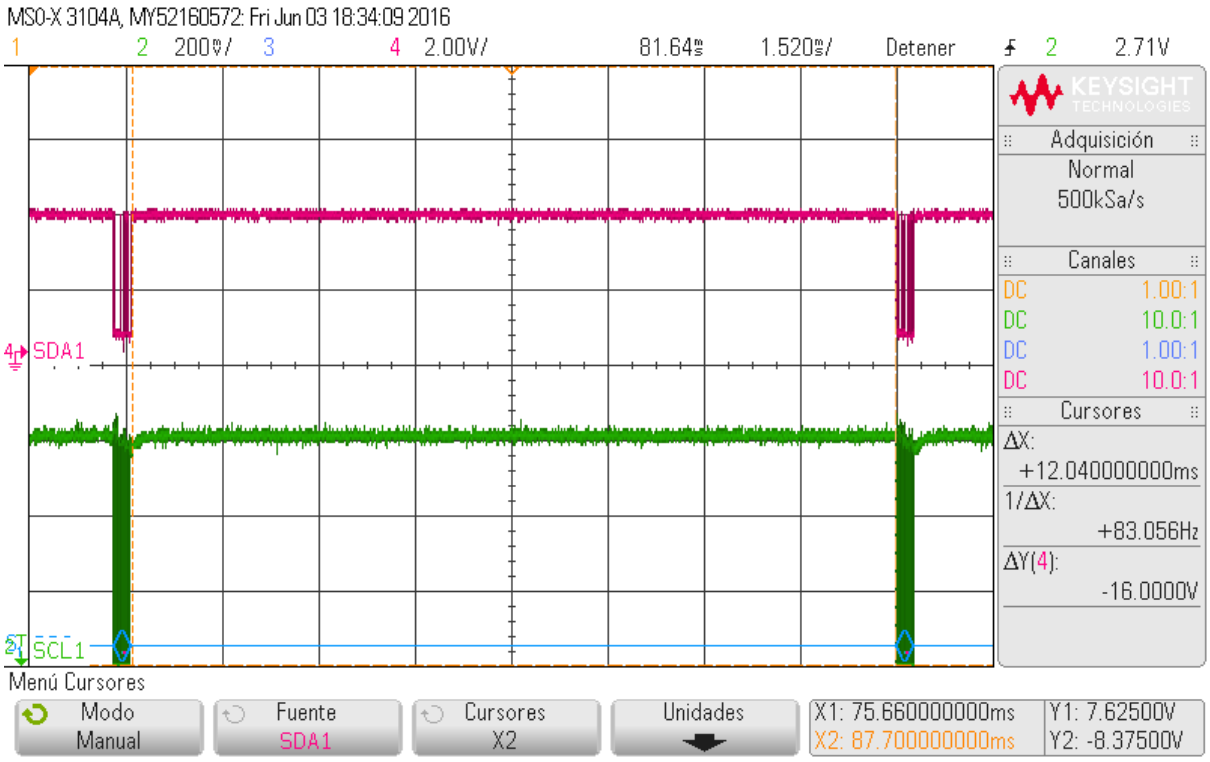

Figura 4.18: Tiempo de espera entre el envío de cada byte

Una vez realizados los experimentos con un único esclavo, se procede a la prueba con una celda completa.

4.6. Escritura de datos en una celda completa

En esta última prueba se procederá a configurar una celda en su totalidad, es decir, se van a enviar las 14 coordenadas a las 19 memorias conectadas a una celda.

Puesto que no se tienen conectadas 19 memorias en la celda, se emulará esa cantidad de dispositivos alternando entre dos de las memorias conectadas.

Al igual que en el caso anterior, se utilizará el paquete software desarrollado, adaptado al trabajo con las memorias utilizadas.

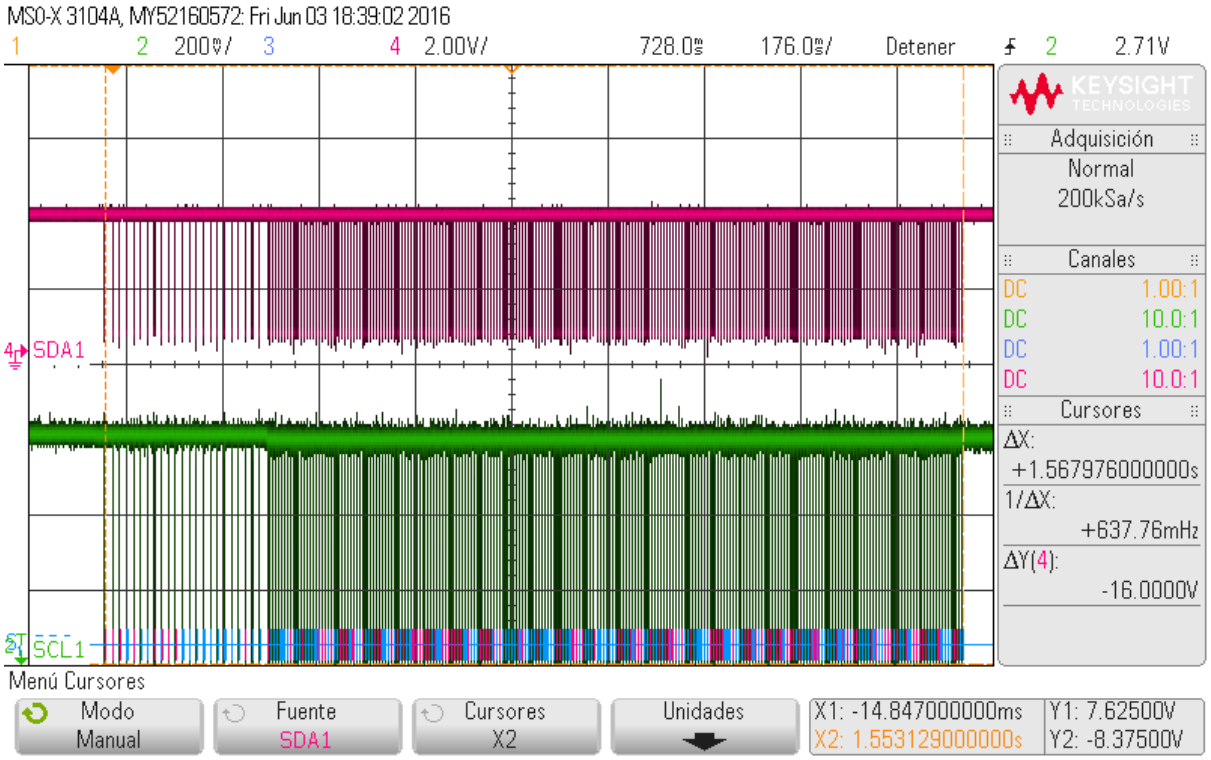

Figura 4.19: Configuración de una celda en su totalidad

Como puede observarse en la captura del osciloscopio, el tiempo total invertido en configurar una celda por completo es de 1.568 segundos.

Sin embargo cabe destacar que este tiempo no corresponde únicamente a la transmisión de los datos, que es lo que interesa, sino que hay que tener en cuenta el tiempo empleado en el guardado del dato y el procesamiento interno. En la prueba anterior se calculó que ese tiempo representa el 96.06 % del tiempo total invertido en el proceso.

Con estos datos se obtiene que el tiempo empleado únicamente en la transmisión de los datos ha sido de 0.061s. También hay que mencionar que no todos los datos transmitidos han sido las coordenadas enviadas, sino que parte de estos datos han sido los propios comandos de escritura y una breve pausa existente, tal como se demostró en la prueba **5**. Esto representa el 32.95 % del tiempo.

Aplicando esta corrección se tiene que las coordenadas han sido transmitidas en un tiempo de 0.04s.

Por último cabe destacar que, al igual que en la prueba **1**, al enviar un volumen de datos más considerable, entran en escena tiempos de pausas no previstas debido al procesamiento interno de la placa de control *BeagleBone*, pausas que son más apreciables al principio de la operación realizada.

No se ha realizado la prueba de configuración de una sección por completo pues se puede extrapolar con los datos de configuración de MUX y las salidas del mismo, pruebas **2** y **4**, y con la configuración de una celda en su totalidad, prueba **7**.

4.7. Resumen de pruebas

Mediante el Cuadro 4.1 y los resultados obtenidos en las pruebas realizadas, se procede a construir una tabla comparativa entre resultados teóricos y prácticos.

$Tarea$	$T_{teorico}(s)$	$T_{practico}(s)$	$Desviacion(s)$
Configuración camino	180×10^{-6}	$201{,}5 \times 10^{-6}$	$21{,}5 \times 10^{-6}$
Envío 2 coordenadas	180×10^{-6}	$192{,}5 \times 10^{-6}$	$12{,}5 \times 10^{-6}$
Envío 14 coordenadas	1260×10^{-6}	4109×10^{-6}	2849×10^{-6}
Envío celda completa	$23,94 \times 10^{-3}$	40×10^{-3}	$16{,}06 \times 10^{-3}$
Envío sección completa	0,65	1,09	0,44
Lectura de un byte	180×10^{-6}	205×10^{-6}	25×10^{-6}

Cuadro 4.2: Tiempos estimados en cada tarea trabajando a 100KHz

Se puede observar que el sistema responde según lo esperado cuando se trabaja con cantidades de información pequeñas, como es el caso del establecimiento de un camino, el envío de una pareja de coordenadas o lectura de un registro.

Sin embargo cuando se pasa a trabajar con una cantidad mayor de datos, como es el caso de las 14 coordenadas para un actuador o el relleno de una celda por completo, se produce una gran desviación entre el valor teórico y el obtenido experimentalmente.

La razón de esto pueden ser las interrupciones que realiza la placa de control *BeagleBone*, tal como se ha visto en las pruebas **1** y **6**, durante las que el BUS I^2C queda pausado.

Adaptando la tabla anterior a la frecuencia propuesta inicialmente, 400KHz, obtenemos los siguientes resultados:

$Tarea$	$T_{teorico}(s)$	$T_{practico}(s)$	$Desviacion(s)$
Configuración camino	45×10^{-6}	$50{,}38 \times 10^{-6}$	$5{,}38 \times 10^{-6}$
Envío 2 coordenadas	45×10^{-6}	$48{,}13 \times 10^{-6}$	$3{,}13 \times 10^{-6}$
Envío 14 coordenadas	315×10^{-6}	$1027{,}25 \times 10^{-6}$	$712{,}25 \times 10^{-6}$
Envío celda completa	$5{,}985 \times 10^{-3}$	10×10^{-3}	$4{,}015 \times 10^{-3}$
Envío sección completa	0,162	0,273	0,111
Lectura de un byte	45×10^{-6}	$51{,}25 \times 10^{-6}$	$6{,}25 \times 10^{-6}$

Cuadro 4.3: Tiempos estimados en cada tarea trabajando a 400KHz

Puede observarse que, a pesar de la diferencia entre los valores teóricos y prácticos, el sistema sigue estando dentro de los márgenes temporales requeridos.

Con esto se da por terminado el banco de pruebas realizado a la placa de comunicación desarrollada y se procede a presentar las conclusiones en el siguiente capítulo.

5

Conclusiones y trabajo futuro

El final del camino está donde te cansaste de andar.

Anónimo

Este último capítulo presenta las conclusiones
Obtenidas de los experimentos realizados,
Así como sugiere formas de mejorar el proyecto.

Tras la realización de los experimentos con el sistema desarrollado y la comparación de los resultados prácticos con los teóricos, se llega a la conclusión de que el sistema no responde de forma esperada ante un entorno en el que debe manejar una gran cantidad de datos.

Tal como se ha podido observar, para comunicaciones sencillas como el configurar un camino hacia una celda, enviar una pareja de coordenadas o leer un registro del actuador, los tiempos teórico y práctico son muy similares. Sin embargo cuando se trata de enviar las siete parejas de coordenadas a un actuador o configurar una celda por completo, se presenta el problema de las interrupciones del procesador de la placa de control *BeagleBone*, lo que hace que la diferencia entre los tiempos teórico y práctico difiera de una forma mucho mayor.

A pesar de esta problemática, el sistema cumple con su propósito, transmitir las coordenadas desde un ordenador central hasta una nube de actuadores, agrupados en celdas, en un tiempo inferior a un segundo.

Como pasos futuros se propone:

- Mejorar la configuración de la placa de control, permitiendo alcanzar la frecuencia propuesta para el BUS I^2C de 400KHz o incluso de 1MHz
- Controlar las interrupciones generadas por la placa de control para que no influyan de forma tan significativa
- Mejorar la implementación software que gestiona la comunicación del BUS I^2C
- Desarrollar un sistema que permita a los esclavos realizar interrupciones en caso de necesitar alertar sobre un problema

Con esto se da por finalizado el presente proyecto.

6

Bibliografía

Por el grosor del polvo en los libros
de una biblioteca pública puede
medirse la cultura de un pueblo.

John Steinbeck

- Mastering the I^2C BUS, por Vincent Himpe, Editorial Publitronic-Elektor, ISBN 090570598X
- Ethernet: The definitive Guide, por Charles E. Spurgeon y Joann Zimmerman, Editorial O'Reilly, ISBN 1449361846
- Exploring BeagleBone, por Derek Molloy, Editorial Wiley, ISBN 1118935128
- Transmisión de datos y redes de comunicaciones, por Behrouz A. Forouzan, Editorial McGrawHill, ISBN 844815617X
- http://www.robot-electronics.co.uk/i2c-tutorial
- https://www.arduino.cc/en/Tutorial/MasterReader
- http://derekmolloy.ie/beaglebone/beaglebone-an-i2c-tutorial-interfacing-to-a-bma180-accelerometer/
- http://makezine.com/magazine/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/
- http://elinux.org/Interfacing_with_I2C_Devices
- http://beaglebone.cameon.net/home/i2c-devices
- https://www.raspberrypi.org/forums/viewtopic.php?f=44t=63345

[1] Proyecto Fin de Carrera de Nasib Fahim Fernández:

http://arantxa.ii.uam.es/ jms/pfcsteleco/lecturas/20130709NasibFahimFernandez.pdf

[2] Proyecto Fin de Carrera de Jesús Castro Murillas:

http://arantxa.ii.uam.es/ jms/pfcsteleco/lecturas/20130617JesusCastroMurillas.pdf

A

Artículo

[1]

[1]Karim Kaci[1], Guillermo Glez-de-Rivera[1], Fernándo López-Colino[1], M. Sofía Martínez-García[1], José L. Masa[1], Javier Garrido[1], Justo Sánchez[2], Francisco Prada[2 3]. Communication Architecture System for Fiber positioning of DESI Spectrograph. Advanced in Optical and Mechanical Technologies for Telescopes and Instrumentation, SPIE. 26/June - 1/July 2016, Edinburgh, UK

Communication Architecture System for Fiber positioning of DESI Spectrograph

Karim Kaci[a], Guillermo Glez-de-Rivera*[a], Fernando Lopez-Colino[a], M. Sofia Martinez-Garcia[a], Jose L. Masa[a], Javier Garrido[a], Justo Sanchez[b], Francisco Prada[bc].
[a]Escuela Politécnica Superior. Universidad Autónoma de Madrid, Madrid, Spain
[b]Instituto Astrofisico de Andalucia, CSIC, Granada, Spain
[c]Instituto de Fisica Teorica, Campus de Excelencia UAM-CSIC, Madrid, Spain

ABSTRACT

This paper presents a design proposal for controlling the five thousand fiber positioners within the focal plate of the DESI instrument. Each of these positioners is a robot which allows positioning its optic fiber with a resolution within the range of few microns. The high number and density of these robots poses a challenge for handling the communication from a central control device to each of these five thousand. Furthermore, an additional restriction applies as the required time to communicate to every robot of its position must be smaller than a second. Additionally. a low energy consumption profile is also desired.
Both wireless and wired communication protocols have been evaluated, proposing single-technology-based architectures and hybrid ones (a combination of them). Among the wireless solutions, ZigBee and CyFi have been considered. Using simulation tools these wireless protocols have been discarded as they do not allow an efficient communication. The studied wired protocols comprise I2C, CAN and Ethernet.
The best solution found is a hybrid multilayer architecture combining both Ethernet and I2C. A 100 Mbps Ethernet based network is used to communicate the central control unit with ten management boards. Each of these boards is a low-cost, low-power embedded device that manages a thirty six degrees sector of the sensing plate. Each of these boards receives the positioning data for five hundred robots and communicate with each one through a fast mode plus I2C bus. This proposal allows to communicate the positioning information for all five thousand robots in 350 ms total.

Keywords: Communication system, Embedded system, Fiber positioner, Spectrograph

1 INTRODUCTION

The international community wants to get a better understanding of the cosmos and the observable universe. With the advancement of technology many research institutions over the world have therefore initiated the creation of the greater spectrographic mapping of the universe. To do that, high-efficient survey spectrographic is becoming more and more necessary to face new challenges in astronomy and physical cosmology. Such instruments are dedicated to observing massive portions of the sky. These must be as efficient as possible in order to minimize inherent managing delay, allowing to achieving the largest possible number of simultaneously observed objects.

The most versatile type of multiobject instruments is the fiber-fed approach, whose primordial and common problem is the positioning of the fiber ends. These must match the position of the objects in a sky target field. For each field to be observed, the configuration of the fibers is different. Therefore, their position must be changed depending on the specific target. The most efficient devices for this task are the actuator arrays, which are able to position all the fiber heads simultaneously; making the reconfiguration time extremely short (typically less than 1 minute) and increasing the efficiency of the instrument. Each of these positioners is an embedded device, a robot, which allows positioning its optic fiber with a resolution within the range of a few microns. The high number, usually thousands, and density of these robots, usually located within a two-meter-diameter circle, pose a challenge for handling the communication from a central control device to each of these positioners.

Several projects have used the fiber positioner-based approach to achieve their objective. An important example is the Cobra fiber positioner developed for the Prime Focus Spectrograph (PFS) instrument on Mauna Kea, Hawaii. It uses 2400 robots to position their respective fibers to be analysed simultaneously with a spectrograph. The Cobra positioner required newly developed piezo tube motors in order to position the optical fiber within 5 µm of its target. It is a two degree of freedom mechanism, a theta-phi style positioner containing two rotary piezo tube motors with one offset from the other, which enables the optic fiber to be placed anywhere in a small circular patrol region [1]. Other well-known case is the Large Sky Area Multi-Object Fiber Spectroscopy Telescope (LAMOST), the convex focal plane of the telescope (diameter 1.75 m) is divided into about 4000 individual domains. Each domain is specified with the position of its fiber with polar coordinates by using its centre point as reference. The 2D positioning error of any fiber should be less than 40 µm and the time required for positioning all the 4000 domains at one time should be shorter than 10 minutes [2, 3]. Also the Echidna multi-object fiber positioner is part of the Fiber Multi-Object Spectrograph (FMOS) project for the prime focus of the Subaru telescope. The focal plane has a diameter of 150 mm the number of fiber is 4000. Each spine is able to position the fiber tip to within 10 µm [4].

In this way, the "Campus de Excelencia Internacional UAM- CSIC" research, as part of his involvement in DESI, has designed and proposed asolution for controlling the five thousand fiber positioners within the DESI focal plane [5, 6] which is presented in this work. The developed positioners can place the fibers with an accuracy of 5 µm, within a maximum time of 16 s and with very low consumption levels. The required fiber positioning time in this project is specified to be around 60 s and the Telescope slew and guide lock < 60 s [7-9]. The challenge is to achieve a bidirectional communication between these thousands of positioners to a central node, to provide the coordinates to each of these positioners and to retrieve their status. Additionally, a low energy consumption profile is always preferred. Based on the experience of this project, our research group has made a comparison and evaluation between several embedded-system-oriented communication protocols. Finally, a communication architecture has been proposed and experimentally evaluatedfor the DESI project. However, it can be easily extended and adapted to others telescopes and missions with the same specifications and scenarios.

The paper is organized as follows. In Section 2, a descriptive study of different technology solutions are presented, finally selecting the most appropriate. The proposed communication solution is described in Section 3, and performance and test results are described in Section 4. The conclusion of this work is presented in Section 5.

2 COMUNICATION SYSTEM: STUDY OF TECHNOLOGIES

The communication architecture has to solve the transmission of the positioning information from a central node (usually a personal computer) to each of the thousands of fiber positioners. It must also allow to retrieving information from each of these final devices. During the description of the different architectures, we will denote as "Master Controller" the central computer that controls all the communication system (fist level); the robot fiber positioners or actuators are presented as the "Final Devices"; finally any intermediate element between the former two will be referred as "Control Devices". The "Final Devices" are grouped in hexagonal-shaped cells of a maximum of 19 positioners. Although it is not a requirement, these cells have been considered as indivisible communication units when defining the different architectures.

The positioners define a coordinate using four bytes. However, to avoid collisions between adjoining positioners, a specific algorithm has to be considered [5]. This is achieved by providing the positioner with a sequence of different coordinates to achieve the final position in a way that avoids these collisions. The worst-case scenario requires no more than sixteen intermediate coordinates, leading to a 64 bytes message.

The configuration task of the communication system consists of two steps. In the first step all the devices will be provided with their position coordinates and allowed to move. In the second step it is detected whether all the devices have been positioned correctly, if not, any wrong positioned device will be reset and re-positioned. This step will be repeated until all devices are properly positioned. Providing initial coordinates to the positioners is the most demanding part of the process. Different embedded-system-oriented communication technologies will be studied focussing on this aspect without forgetting that a bidirectional communications has to be attainable:

- Wireless:
  - Zigbee: It is a wireless communication protocol; it is designed for low-speed communications between two or more devices. It is possible to form networks with thousands of devices communicating to each other [10].
  - CyFi: CyFi is a 2.4GHz ISM radio transceiver, which is able to support data rates up to 1 Mbps radio. CyFi has a 1-Mbps GFSK, packet data buffering, packet framer, DSSS and RSSI [11].
- Wired:
  - I^2C: It is a serial synchronous communication protocol that uses two ground-referred bus lines to transmit information [12].
  - CAN: A serial asynchronous protocol that uses a differential bus for the transmission of messages in a distributed environment [13].
  - Ethernet: An IEEE communication protocol 802.3 standard [14].

The following sections will discuss different architectures to solve the communication problem. First ones will present single technology-based solutions. Later, hybrid proposals will be described.

2.1 Wireless

The main problem of the wireless solutions, observed when studying a solution using Zigbee, is derived from a high density of antennas (when the number of antennas is over 5,000 in a small environment). A simulation was performed to analyse the approach, which resulted in the impossibility of this solution due to the high distortion that would cause the antennas to each other for receiving information [15].
As seen in Figure 1, the proposed architecture would be tree architecture, where each Control Device would act as a zigbee router able to reach 250 Final Devices. The approximate configuration time would be about 10.82 s.

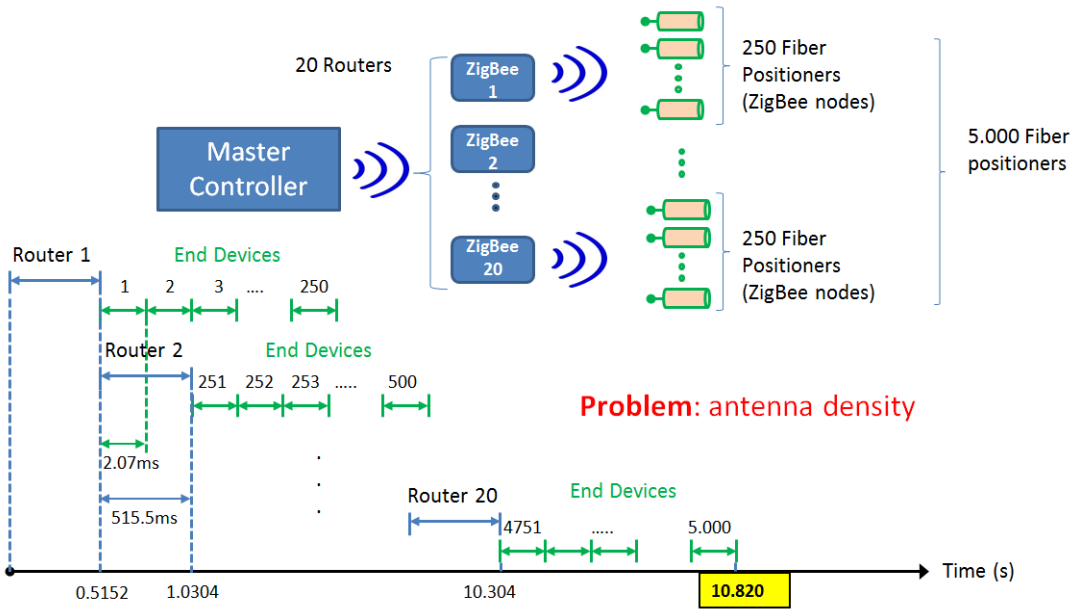

Figure 1. Architecture and delay using Zigbee

2.2 Wired

After the impossibility of an architecture using wireless technology, different wired solutions were studied. The transmission time, the advantages and disadvantages of the I^2C and CAN bus technologies were analysed, making a comparison between both.

- I^2C: This technology uses two bus lines, one to send data (SDA) and the other as clock signal (SCL). It is a half-duplex protocol with a master-slave relation between the different nodes in the bus. The master always starts the communication addressing one slave and sending or requesting data. With a tree type architecture all Final Devices would be configured in an approximated time of 4.6 s. The architecture has three levels: the first level with four Control Devices, each controlling another 65 Control Devices in the second level, each of them will manage a cell of 19 Final Devices, which define the third level. Each level will operate as a router for the next level. The main problem of this structure is that it would have 260 Control Devices, which would imply a lot of hardware, increasing the problem of consumption. Another problem is that the I^2C bus between the Master Controller and the first level of the architecture there is a bottleneck, which increases significantly the configuration time of all the Final Devices. Figure 2 shows this proposed architecture.

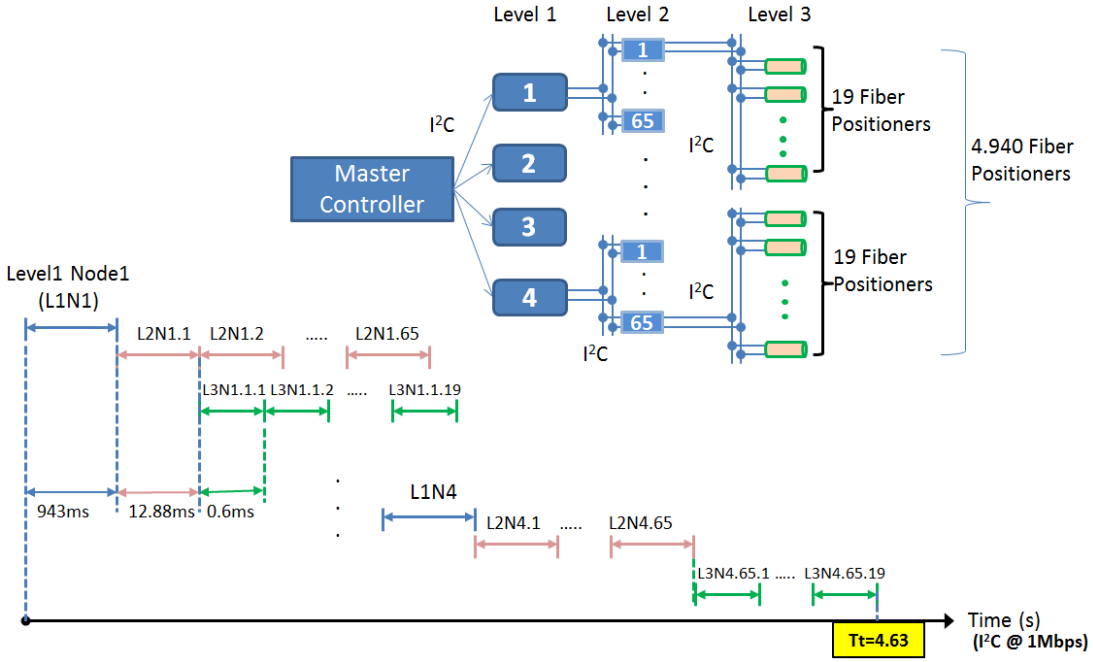

Figure 2. Architecture and delay using I^2C

- CAN: This is a Multi-Master broadcast half-duplex serial bus technology in which all nodes may receive and send data. A tree type architecture may configure all final devices in an approximate time of 7.69 s. The proposed architecture has two levels, the first level with 260 Control Devices, which would control a cell device. As in the previous approach the problem is that it would have 260 control boards, increasing the power consumption. Also, it must be noted that CAN protocol requires more energy than the equivalent I^2C-based solution. The CAN based proposal is depicted in Figure 3.

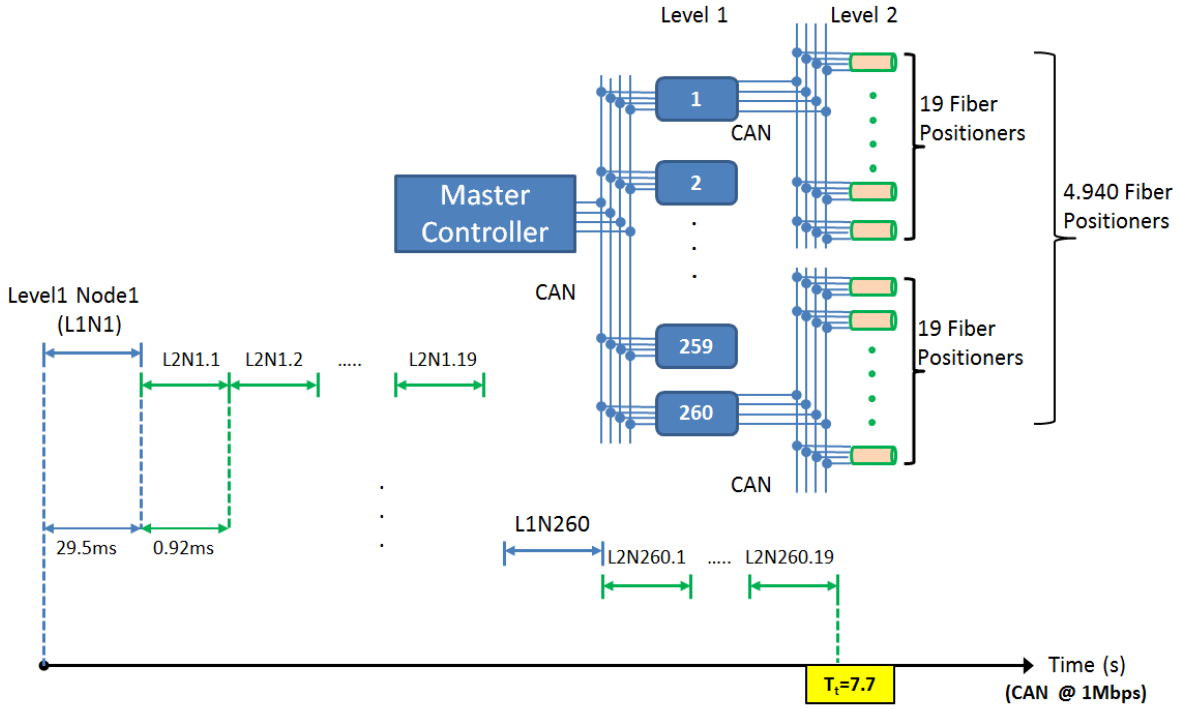

Figure 3. Architecture and delay using Bus CAN

Finally, Table 1 resumes different characteristics of these protocols. This information is quite relevant to select one of them for the final solution.

Table 1. Comparative I^2C and Bus Can.

	CAN Bus	**I^2C Bus**
Speed	Up to 1Mbps	Up to 3.4 Mbps
Acknowledgement	Yes	Yes
Error Detection	Yes	No
Unlimited data per frame	No	No
Differential bus	Yes	No
Slave interruptions	Yes	No
Automatic retransmission	Yes	No
CSMA-CD	Yes	No
Processor Load	High	Low
Energy Consumption	High	Low

2.3 Hybrid

As the solution is unviable with wireless technologies due to the high density of antennas and the consumption and hardware complexity is evident with the wired solution, a mixed design is studied.

- CyFi + I^2C: CyFi is a Wireless technology able to transmit at 1 Mbps. Provides security and less consumption than Zigbee. The central unit would be connected to CyFi to 250 Control Devices, which would configure the devices via I^2C. The problem is however the same because of consumption of the high-density hardware. Figure 4 depicts this approach.

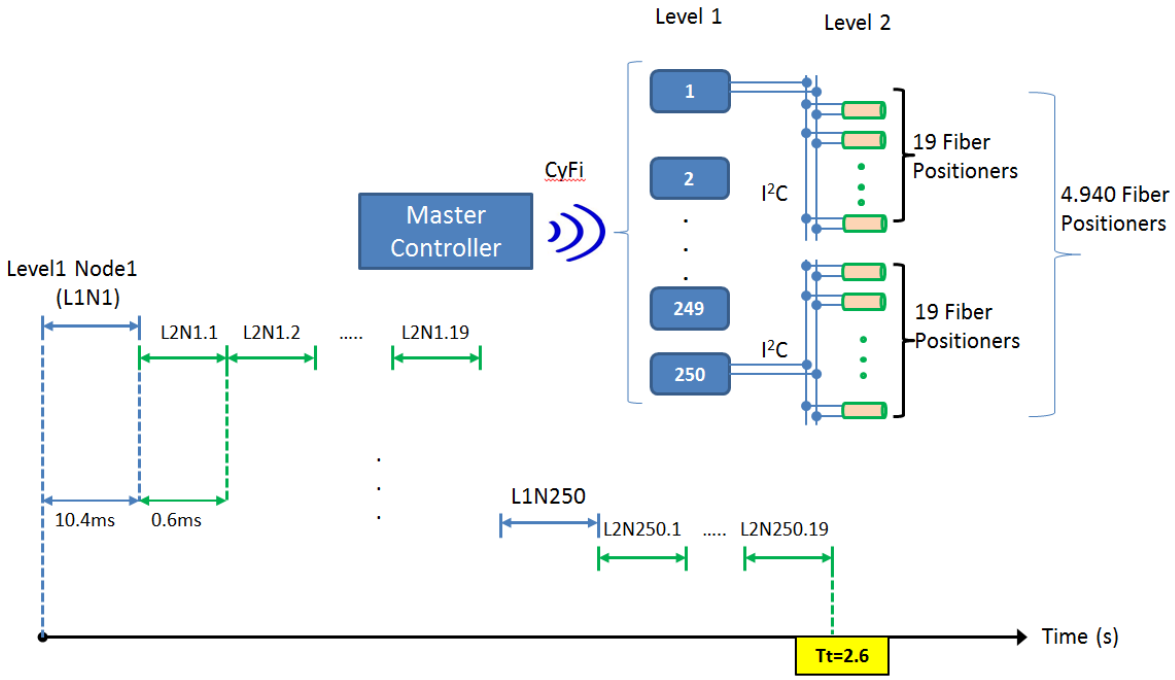

Figure 4. Architecture and delay using CyFi+I2C.

The study of the estimated time for providing the full set of positions to the 5000 Final Devices according to each of the different architectures is presented in Table 2.

Table 2. Required time for configuring 5000 Final Devices using the different architectures.

Technology	**Time (s)**
Zigbee	10.82
Bus CAN @ 1 Mbps	7.69
I^2C @ 1Mbps	4.63
CyFi +I^2C	2.61

In all the proposed solutions, the configuration time of the devices increases mainly due to the bottleneck from the central unit to the first level. Including the Ethernet protocol allows us to send information securely up to 10 Gbps on a middle range, so it is a candidate to replace the first level link, where the bottle neck is allocated.

3 PROPOSED SOLUTION

Given the difficulty of incorporating wireless solutions, the bottle neck caused by the I^2C link between the Master Controller and the Control Devices and the high number of Control Devices of the other architectures, our proposal is a solution combining Ethernet and I^2C protocol.

The telescope's focal plane is logically divided into ten sectors (see Figure 5). Each of these sectors will have one Control Device to forward the configuration information to the 500 Final Devices, fiber positioners, of that sector. In each sector the fiber positioners are grouped into cells of 19 Final Devices. Each cell is connected to a fast mode plus I^2C bus achieving transfer rates up to 1 Mbps. Considering that a single I^2C bus can handle up to 127 devices, it is not possible to use the same I^2C bus to communicate all the positioners with their Control Device. Each of the cells will be connected to an output of a set of I^2C multiplexers. The Control Device, acting as master of the I^2C communication, will configure the corresponding multiplexer to a particular output to exclusively reach to each cell. This approach has two advantages: first, the reutilization of I^2C address for each cell, simplifying its management; second, reducing the global capacitance of the bus as only few elements are connected simultaneously to the bus. On the other hand, to avoid the

existing bottle neck in the first level, this link is replaced by an Ethernet connection that allows transferences up to 10 Gbps, so the configuration time of all Controller Devices is reduced; making a standard interface between the Master Controller and the first I^2C components. This structure allows different load distributions, in order to respect the focal plate structure.

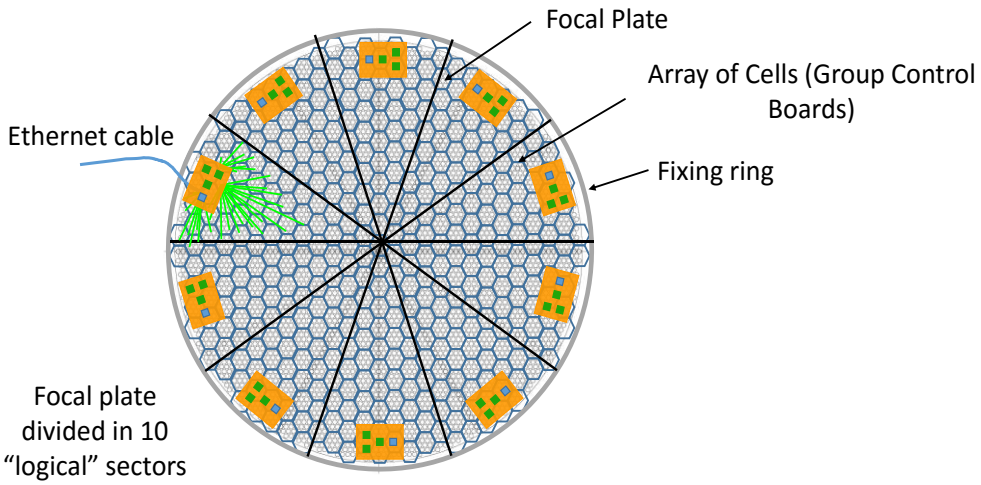

Figure 5. Focal plane.

As it has been said, from the Master Controller ten Control Devices will be connected using Ethernet. The Control Device must have both Ethernet and I^2C connectivity. Between each Control Device and the Final Device there are four eight-output I^2C multiplexers. Using this configuration, each multiplexer output allows connecting up to 123 devices. Therefore, Control Devices can theoretically reach to 3936 (123×4×8) Final Devices each. Therefore, using this configuration is possible to reach to 39360 Final Devices. These numbers clearly exceeds the requirement of this telescope. This proposal lowers the hardware density, reducing time and power consumption.

Figure 6 shows the overall communication proposal followed up today's. This proposal allows the Master Controller to link with all the positioners distributing the workload among the two routing levels. It has been designed with a low power consumption philosophy, in order to minimise the thermal noise in the focal plane.
The communication system consist of a common PC as Master Controller; in the first routing level, the Control Devices are Raspberry Pi 3 model B, then I^2C 8-chanel multiplexers PCA9847 from NXP Semiconductors. Each I^2C output in the multiplexer is connected to 19 Final Device.

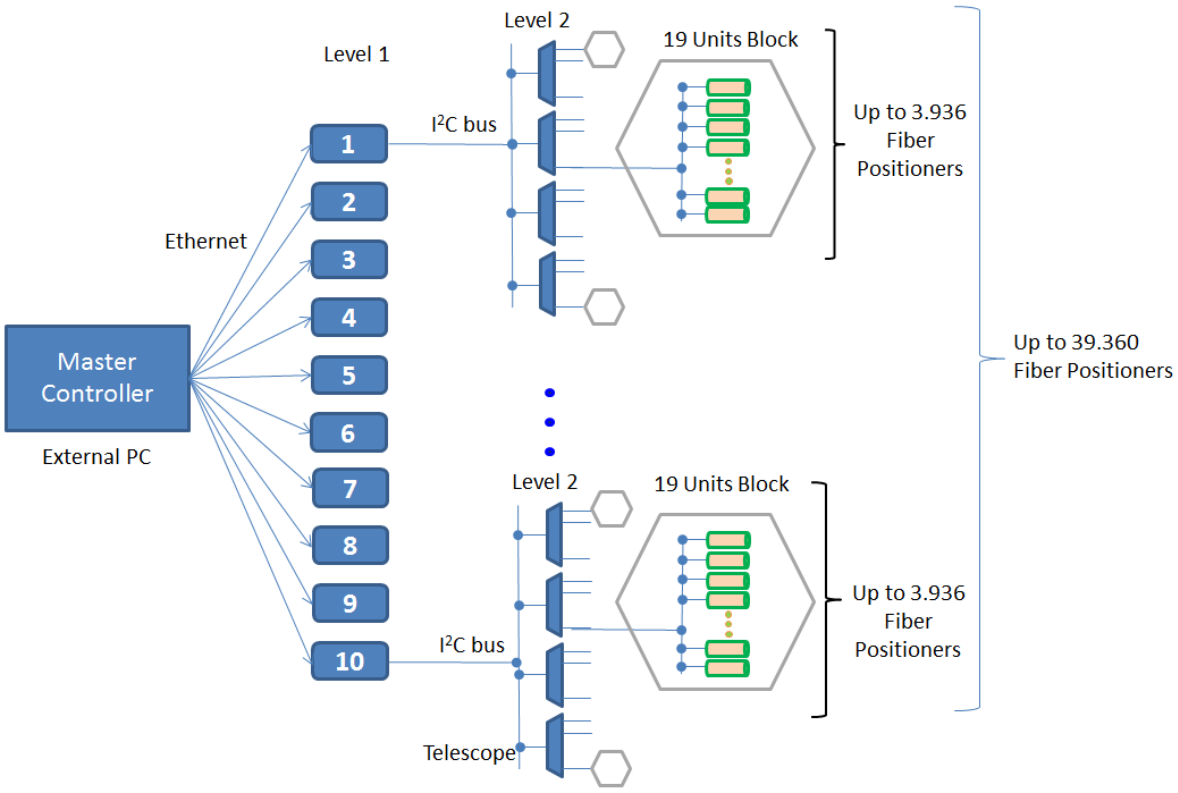

Figure 6. Architecture using Ethernet+I²C.

4 EXPERIMENTAL RESULTS

For the proposed solution, several performance tests have been carried out focusing on the communication time between the Master Controller and Final Devices, and the power consumption of the communication architecture. First, the results of the communication time between the different levels of the architecture will be presented and explained. Next, the power consumption levels of the different elements of the proposal will be shown.

4.1 Transmission time

To measure the transmission time, the definition of the transmitted information is required. As it was presented in Section 2, the worst case scenario requires sending 64 bytes to each of the Final Devices. This data represents a sequence of sixteen sequential coordinates required to reach to the final position without colliding with any adjacent Final Device. It is possible to add an extra byte at the beginning of the data to inform the Final Device of the number of steps that has to perform. This byte is clearly an overhead, but it grants both a simple information verification and variable data length. Therefore, it may justify its inclusion. It is also possible to include a two-byte address[1] to inform the Control Device of the addressee of the following data frame. The architecture can be programmed to fulfil its task without this overhead but it will show its benefits when addressing a low number of Final Devices. Finally, two fields are required in every Ethernet communication between the Master Device and each of the Control Devices: a byte for defining the action (provide information to the Final Devices, or request information from them) and the length of the whole transmission (4

[1] The Final Device unique address has been defined as follows: five bits to identify which of the 19 devices per cell; three to specify the output of the multiplexer; two bits to identify the multiplexer; finally, four bits to specify the Control Devices There are two spare bits, reserved for future use.

bytes). For a maximum length of 33505 bytes in a single Ethernet communication. Figure 8 depicts the structure of an example transmission including all the presented elements.

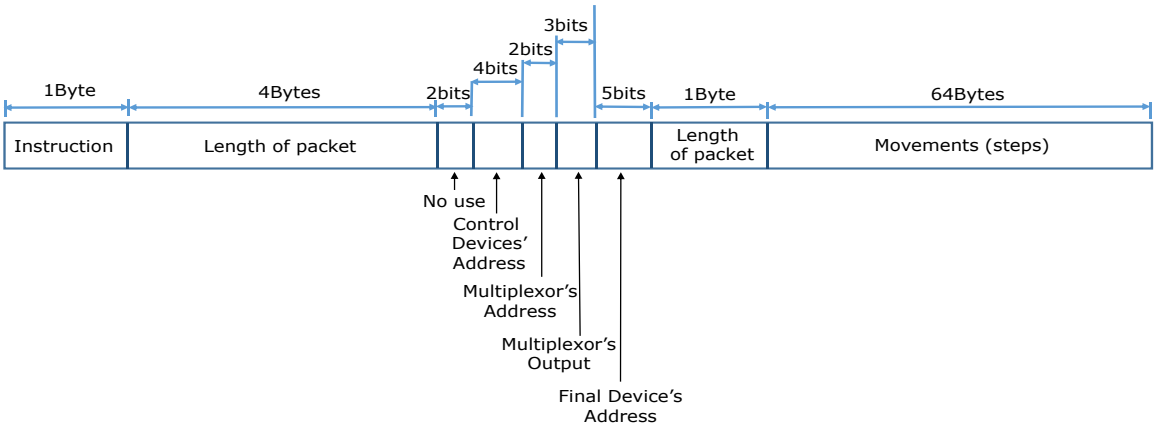

Figure 8. Example data frame sent to a Control Device including information of one Final Device

Another aspect related to the communication time concerns to the Ethernet communication speed. Although the maximum Ethernet transmission rate is 10 Gbps, there is a limitation of the transmission rate due to the embedded device used as Control Device. The Raspberry Pi 3 Ethernet port only allows transmission rates up to 100 Mbps. However, considering that the I^2C transmission rate is 1 Mbps, the limitation to 100Mbps of the Ethernet connection will not suppose any bottle neck in this link.

A study of the different elements included in the data frame (length and address) has been performed. The different combination of elements in the header have been defined as set 1-4 as follows:

- Set 1: The *Basic* frame has a size of 64 bytes.
- Set 2: The *Basic+Address* frame 66 bytes.
- Set 3: The *Basic+Length* frame 65 bytes.
- Set 4: The *Basic+Address+Length* frame, with a total of 67 bytes.

The four sets have been evaluated in four different scenarios resembling different situations that can be present in the communication process:

- Scenario 1: Send to the 500 Final Devices a set of 16 positions (64 bytes)
- Scenario 2: Send to one Final Device a set of 16 positions (64 bytes)
- Scenario 3: Send to the 500 Final Devices a set of 1 position (4 bytes)
- Scenario 4: Send to one Final Device a set of 1 position (4 bytes)

Table 3 presents the required theoretical time to send this information from the Master Controller to a Control Device using a 100 Mbps Ethernet link and a TCP/IP stack. It can be shown that the overhead of the address and the data length makes the set 1 the best solution for the initial massive submission of data to the positioners. However, when the Master Controller addresses a low number of Final Devices or reduces the number of positions, the other sets show better results.

Table 3. Theoretical performance between Master Computer and Control Device (Ethernet Link)

	Set 1 (µs)	**Set 2 (µs)**	**Set 3 (µs)**	**Set 4 (µs)**
Scenario 1	$2.73x10^3$	$2.82x10^3$	$2.78x10^3$	$2.86x10^3$
Scenario 2	$2.73x10^3$	13.12	52.96	13.2
Scenario 3	$2.73x10^3$	$2.82x10^3$	$2.1\ x10^2$	$3.0x10^2$
Scenario 4	$2.73x10^3$	13.1	48.2	8.4

To evaluate the worst-case scenario, an experimental test was performed communicating a PC (Intel Core 2 Duo E8200 2.66 GHz and 6 GB RAM with a Debian GNU/Linux 8 (Jessie) 64 bits) with a Raspberry Pi 3 B (Raspbian Jessie Lite 4.4) and an Ethernet switch (Sky Link Net 1008 100 Mbps). The average transmission time for the scenario 1 and set 4 showed an average transmission time of 3.107 ms, greater than the estimated value. Also, the switching time between two Control Devices was measured to be 896 us. Therefore, the required time to send 16 positions for the 5,000 Final Devices to their respective Control Devices is 10×3.107 ms + 9×896 us = 39.142 ms.

Table 4 presents the required theoretical time that each Control Device would require to transmit the required information to the 500 Final Devices using a 1 Mbps I2C link. It is worth noting that the data length field (present in sets 3 and 4) is the most relevant field for reducing the transmission time.

Table 4. Theoretical performance between Control Device and final devices (I^2C Link)

	Set 1 (µs)	**Set 2 (µs)**	**Set 3 (µs)**	**Set 4 (µs)**
Scenario 1	$2.62x10^5$	$2.62x10^5$	$2.66x10^5$	$2.66x10^5$
Scenario 2	$2.62x10^5$	$5.24x10^2$	$5.32\ x10^2$	$5.32\ x10^2$
Scenario 3	$2.62x10^5$	$2.62x10^5$	$2.60\ x10^4$	$2.60\ x10^4$
Scenario 4	$2.62x10^5$	$5.24x10^2$	52	52

To evaluate the real time that Scenario 1 would require using sets 3 or 4 a laboratory test has been performed. In this case, the Final Device have been implemented using MBED LPC1768 and the Control Device was the same Raspberry Pi 3 B as used in the previous experiment. For this evaluation an over clocked 1.2 Mbps I^2C link has been stablished between these elements. This experiment aimed to measure the average time required for the three main steps in the I^2C stage of the communication: the time required to send the 65 bytes to a whole cell, the time required to configure a single multiplexer to swap to another output and the time required to swap between multiplexers. It is worth noting that only seven of the multiplexer's outputs are used. Table 5 presents these measures and the required time for a Control Device to fulfil the Scenario 1 and Set 3-4.

Table 5. Experimentally measured times (I^2C Link)

Measure	**Average Time (µs)**	**Std. Deviation (µs)**
Fill a 19-elements cell (65 bytes each)	$1.10x10^4$	23.0
Mux switching output	42.5	1.5
Switching mux	42.5	1.5
Filling 513 Final Devices (65 bytes each)	$3.10\ x10^5$	30.0

The greater time observed in the experimental results when evaluating the scenario 1 and set 4 is justified because of the required configuration time of the multiplexers and a delay introduced by the Raspberry between consecutive I^2C packages.

Considering the hierarchical structure of the proposed architecture, the time used in the I^2C link (310 ms) is the bottleneck of the approach, therefor the time of the Ethernet link (39 ms) is much lower. However, considering that the ten Control Devices perform their tasks in parallel, the average time required to communicate a full set of coordinates to 5,130 Final Devices is 349.78 ms.

4.2 Power consumption

The former experiments have allowed us to measure the power consumption of a single sector communication architecture comprising a Control Device (Raspberry Pi 3 B) and a set of four multiplexers. The average amount of power is 1.605 W at 5 V of power supply: the Raspberry requires 1.52 W and the multiplexers 0.085 W. The whole communication architecture would require 16 W. However, the communication system can be switched to a low power mode (or turned off) during the observation period.

5 CONCLUSIONS

In this paper, we have presented a comparison between different communication architectures for solving the communication between a central Master Control and the 5,000 fiber positioners proposal of the DESI project. Due to the high density of antennas required for the full wireless approach, this single technology-based has been discarded. The single-technology wired approaches, I^2C and CAN, were not good candidates considering power consumption and hardware complexity issues. Among the hybrid approaches, the Cyfi-I^2C approach presented the same disadvantages of the former two single-technology based approaches. Finally, the chosen proposal is a hybrid Ethernet-I^2C solution because it shows the best estimated timing performance and reduces both the hardware and power requirements.

For a proof of concept, the hybrid Ethernet-I^2C architecture has been evaluated. This experiment has measured the communication time from the Master Controller to a cell of 19 Final Devices, implemented using 19 MBED LPC 1768 for emulating the fiber robot positioner control devices. The Control Devices have been implemented using embedded platforms connected to four multiplexers. The required time to fill the whole set of 5,000 fiber positioners is lower than 350 ms. The estimated time for updating the position of a single fiber positioner, required for the fiber view camera position correction, is about 100 µs.

ACKNOWLEDGEMENTS

This work was funded by the Spanish MINECO, under the AYA2014-60641-C2-2-P project.

REFERENCES

[1] Fisher, C., Braun, D., Kaluzny, J., and Haran, T., "Cobra: A two-degree of freedom fiber optic positioning mechanism," in 2009 IEEE Aerospace conference, pp. 1-11, (2009).

[2] Xing, X., Zhai, C., Du, H., Li, W., H.Hu, Wang, R., *et al.*, "Parallel controllable optical fiber positioing system for LAMOST," presented at the SPIE Advanced Technology Optical/IR Telescopes VI, (1998).

[3] Zhang, Y. and Qi, Y., "A type of displacement actuator applied on LAMOST," in SPIE 701928-1-701928-5, pp. 701928-1-701928-5, (2008).

[4] Moore, A., Gilligham, P., and Saunders, W., "A 400 multi-fiber system based on teh Echidna Positioner " in Astronomical Society of the Pacific Conference Series, (2008).

[5] Fahim, N., Prada, F., Kneib, J. P., Sánchez, J., Hörler, P., aRillaga, X., *et al.*, "An 8-m diameter Fiber Positioner Robot for Massive Spectroscopy Surveys," Astronomy & Astrophysics manuscripts, vol. 8, October 3, 2014 (2014).

[6] Silber, J., Schenk, C., Anderssen, E., Bebek, C., Becker, F., Besuner, R., *et al.*, "Design and performance of an R-q fiber positioner for the BigBOSS instrument," in SPIE. The International society for optics and photonics pp. 845038-1-845038-13, (2012).

[7] Fahim, N., Rivera, G. G.-d., Castro, A., Garrido, J., Sanchez, J., and Prada, F., "Mechatronics for micrometric optical fiber positioning in a telescope focal plane," in Design of Circuits and Integrated Systems (DCIS), San Sebastian, Spain, (2013).

[8] Schelegel, D., Abdalla, F., Abraham, T., Ahn, C., Prieto, C. A., Annis, J., *et al.*, "The BigBOSS Experiment," arXiv, vol. 1106.1706, (2011).

[9] Schlegel, D., Bebek, C., Heetderks, H., Ho, S., Lampton, M., Levi, M., *et al.*, "BigBOSS: The Ground-Based Stage-IV BAO Experiment Submitted by LBNL and NOAO," in arXiv, (2009).

[10] Alliance, Z. *www.zigbee.org*.

[11] Tomorrow, C. E. i. (2016). *www.cypress.com*.

[12] Bus, I. C. (2016). *www.i2c-bus.org*.

[13] "ISO11992," 2016.

[14] IEEE. *www.ieee802.org/3*.

[15] Campos, J. L. M., "BigBOSS Collaboration Meeting: Preliminary Study Antenna System of a Radio Telescope," 2012.

B

Presupuesto

1) Ejecución Material

- Compra de ordenador personal (Software incluido) — 2.000 €
- Alquiler de impresora láser durante 6 meses — 250 €
- Material de oficina — 150 €
- Total de ejecución material — 2.400 €

2) Gastos generales

- sobre Ejecución Material — 352 €

3) Beneficio Industrial

- sobre Ejecución Material — 132 €

4) Honorarios Proyecto

- 1800 horas a 15 €/ hora — 27000 €

5) Material fungible

- Gastos de impresión — 280 €

- Encuadernación — 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto — 30.364 €

7) I.V.A. aplicable

- 21 % Subtotal Presupuesto — 6376,44 €

8) Total presupuesto

- Total Presupuesto — 36.740,44 €

Madrid, Junio 2016

El Ingeniero Jefe de Proyecto

Fdo.: Karim Kaci

Ingeniero de Telecomunicación

C

Pliego de condiciones

Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un *Sistema de Comunicación con altas restricciones de tiempo y consumo entre un maestro y múltiples esclavos*. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales.

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la

obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a

la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrataz anteriormente llamado "Presupuesto de Ejecución Material"que hoy designa otro concepto.

Condiciones particulares.

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.