9. Anexos

9.1. Diseño del soporte de la cámara.

El hecho de poder cambiar la posición de la cámara sin tener que mover el robot supone un ahorro de energía y un mayor aprovechamiento de la situación del robot en cada instante, de modo que es posible tomar imágenes en varias posiciones.

Por todo ello, ha sido imprescindible diseñar un soporte para la cámara que se emplea. El software empleado en el diseño del soporte ha sido SolidWorks [6] dado que al ser estudiante se facilita una licencia para este producto, además de que se permite crear modelos 3D fácilmente, siendo posible exportarlos posteriormente a formato *.stl*, de modo que se puedan imprimir en cualquier impresora 3D.

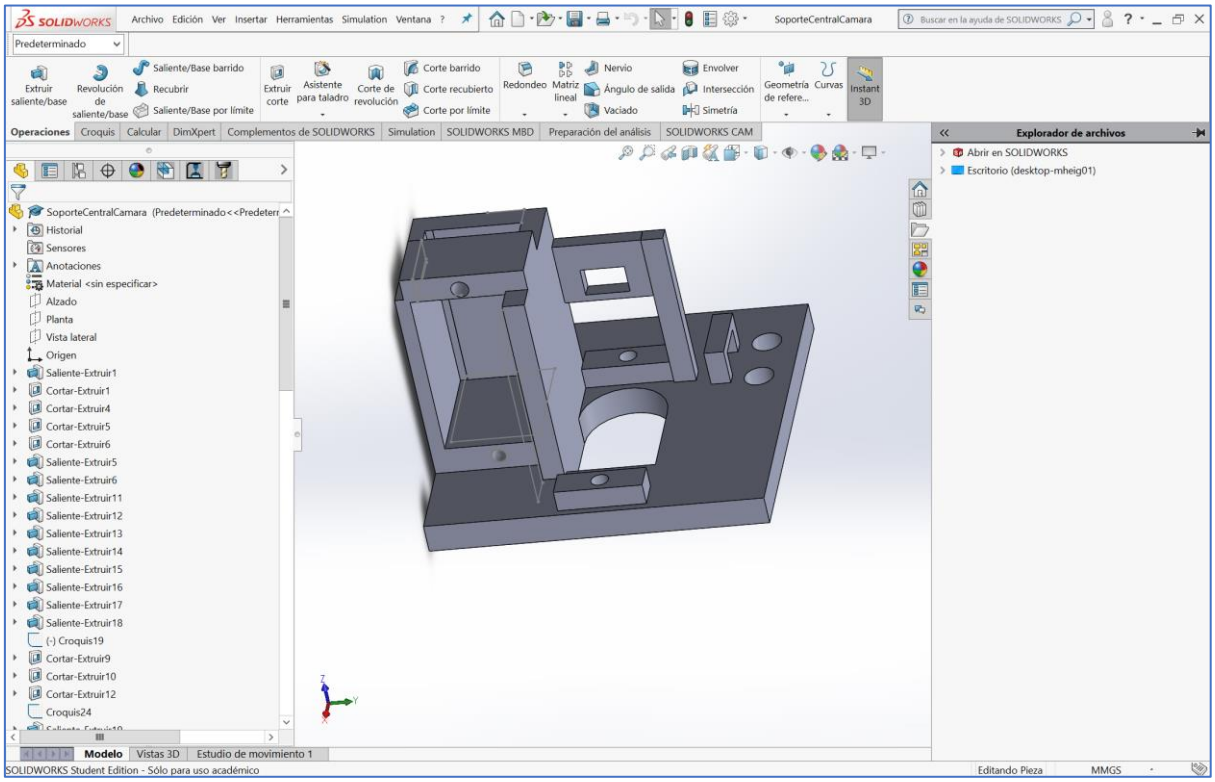


Ilustración 29: Interfaz de SolidWorks.

Para poder realizar el diseño se han tenido en cuenta los siguientes criterios:

- El soporte debe caber en la parte superior del robot Bioloid y la base de este debe de estar sujeto al robot mediante tornillos.

- El objeto en sí debe de albergar dos servomotores para poder ofrecer dos grados de libertad a la cámara. Un servomotor debe encargarse de la rotación de la base y el otro de la rotación del cabezal.
- Para evitar problemas con la longitud del cableado, es necesario que el tamaño sea lo más reducido posible.
- Se deben tener en cuenta los orificios de todos los cables, tanto de los servos como de la cámara.

Con estos criterios, se ha podido crear un diseño dividido en las siguientes piezas:

9.1.1. Base del Robot

Se trata de una pieza de apoyo dado que únicamente tiene como función hacer que la hélice del servo quede fija en ella. Por ello se hace un pequeño orificio con la forma de la hélice y esta se adhiere mediante un adhesivo para plásticos.

Además, esta parte incluye 4 puntos de apoyo para la base del soporte y unos agujeros donde ubicar los tornillos que fijan esta base con el robot.

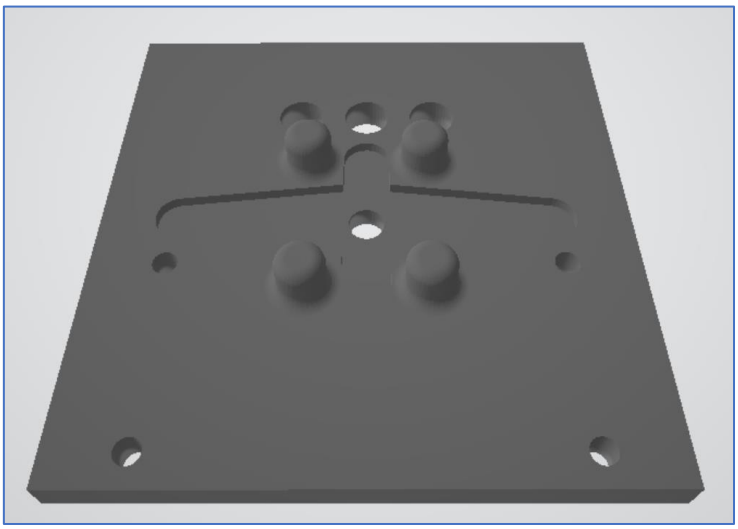


Ilustración 30: Pieza Base del Robot.

9.1.2. Base del Soporte:

En esta parte, se realiza un diseño para poder ubicar los dos servomotores. El modelo empleado de este componente es el SG90 dado que puede funcionar a 6 voltios y su tamaño entra en los rangos usuales del mercado.

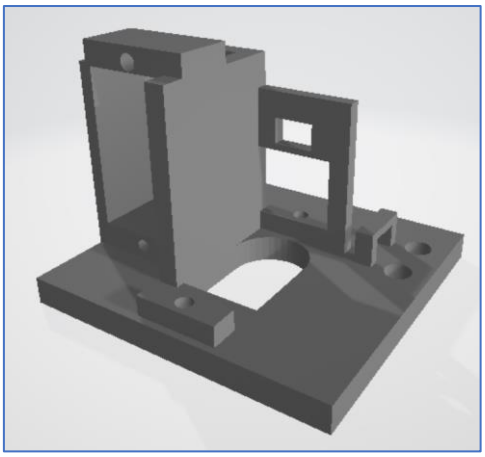


Ilustración 31: Pieza Base del Soporte.

9.1.3. Cabezal

Es el componente encargado de sostener la cámara además de poder realizar el segundo de los dos movimientos producidos en el soporte, de modo que permite inclinar la cámara cuando el momento lo requiera.

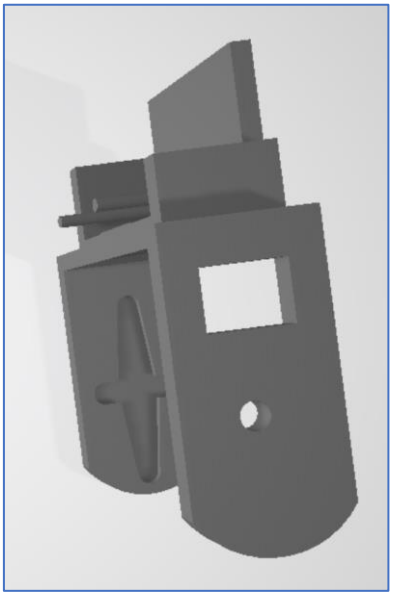


Ilustración 32: Pieza Cabezal

9.1.4. Otros

Además de estas partes principales, se han diseñado complementos para poder asegurar la fijación y estabilidad de todo el soporte, como pueden ser tapones y arandelas.

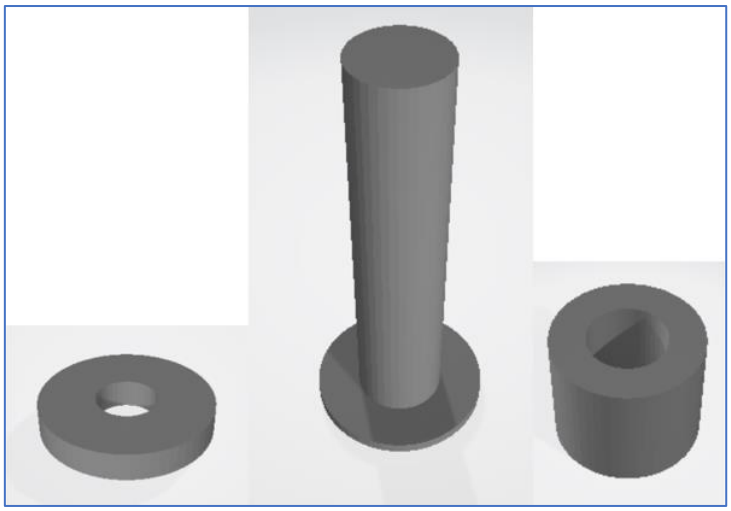


Ilustración 33: Piezas Secundarias.

Al final, se ha podido obtener un soporte para la cámara con el siguiente aspecto:

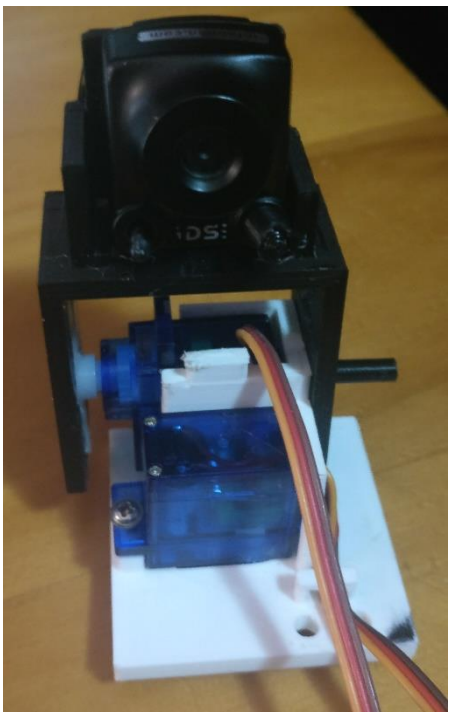


Ilustración 34: Soporte de la cámara.

9.2. Diseño de la cape.

Para poder realizar una captación de valores medibles de aceleración y magnetismo, se ha decidido desarrollar una cape personalizada para la BeagleBone Black. Para ello se ha empleado la herramienta online *EasyEDA* [7], de modo que con ella es posible diseñar un circuito electrónico con todo tipo de componentes.

Cada componente suele tener relacionada una librería de su diseño creada por los desarrolladores de la herramienta o por usuarios de la plataforma. Esta librería permite exportar el circuito finalmente diseñado a un modelo de placa PCB para finalmente poder pedir la producción o fabricación de esta.

El circuito a diseñar mediante esta herramienta tiene como principal objetivo llevar a cabo la sensorización, pero además, debe controlar otras cuestiones relacionadas tanto con el control del robot BIoloid como con la captura de imágenes.

9.2.1. Función de la placa:

- **Sensorizaciones**: El circuito debe llevar incorporado una unidad inercial de medida (IMU) para poder obtener valores medibles de acceleración, magnetismo. Este apartado incluye una comunicación mediante I2C entre el sensor y la Beaglebone para poder obtener dichos valores.
- **Extracción de valores**: Otra parte de gran inportancia se basa en poder sacar los valores de la Beaglebone para que otro dispositivo totalmente ajeno pueda obtenerlos sin problemas. Para ello se ha decidido emplear la tecnología bluetooth. De este modo, se introduce un módulo bluetooth comunicada con la Beaglebone mediante comunicación UART.
- **Control soporte de cámara**: El circuito debe poder controlar este soporte que tiene dos grados de libertad y, por lo tanto, dos servomotores. Para ello se necesitan dos borneras de tres pines cada una, de modo que estén conectadas a varios terminales PWM de modo que cada señal de este tipo pueda controlar un servo.
- **Comunicación Beagle-Bioloid**: Para poder simplificar la comunicación de la Beagle con el robot Bioloid y viceversa, el circuito debe llevar incorporado una bornera de dos pines, de modo que dichos pines lleven a cabo la comunicación serie con el robot.
- **Diseño modo Cape**: El hecho de que este circuito debe de estar directamente conectado con la Beaglebone, obliga a que la placa PCB resultante debe tener forma de cape, teniendo la placa todos los pines dispuestos a ser conectados en la Beagle.

Estas son las funciones necesarias que debe realizar la cape personalizada y a continuación se enumeran los componentes [8]incluidos en ella.

9.2.2. Componentes:

- **Resistencias**: Fabricadas con película de carbono, necesarias para controlar las intensidades y poder crear puentes Pull-Up en el circuito, de modo que se puedan controlar correctamente los LEDs y las configuraciones del módulo Bluetooth y de la IMU mediante señales GPIO.

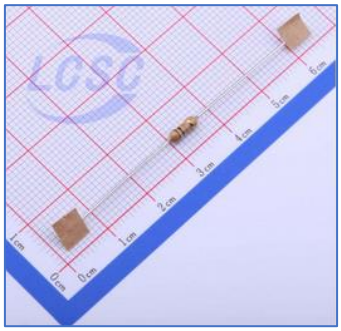


Ilustración 35: Resistencia de 10 KOhmios.

- **Condensadores**: Componentes cerámicos para completar los circuitos de comunicaciones y poder separar varias señales de la masa.

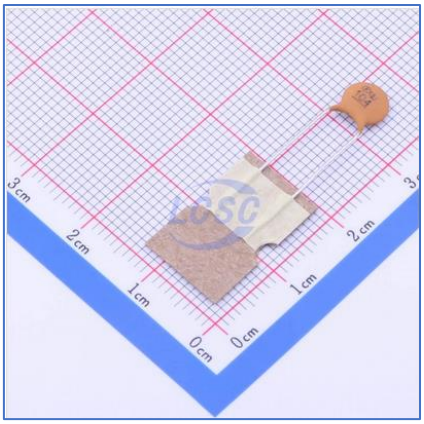


Ilustración 36: Condensador cerámico de 100 nF

- **LEDs**: Necesarios para poder indicar el estado del módulo Bluetooth. Mientras el LED 2 está encendido, el módulo Bluetooth está encendido y

configurándose. Según la frecuencia con la que este LED parpadea, se indica la velocidad de comunicación entre la Beaglebone y el módulo Bluetooth. En cambio, si el LED 1 parpadea, se indica que se recibe o se envía información de publicación entre el módulo y la Beagle.

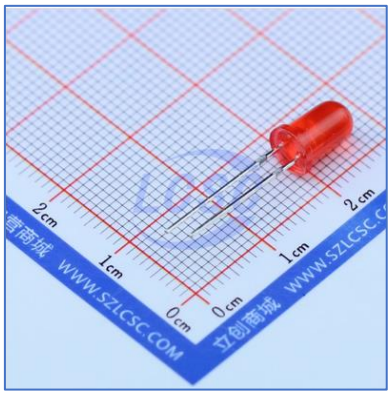


Ilustración 37: LED Rojo

- **Borneras**: Son necesarias para poder conectar los dos servos que mueven el soporte de la cámara y para poder comunicar la Beagle con el Robot. Las borneras de tres terminales son para controlar los servos. La bornera de dos terminales se utiliza para comunicarse con el Robot Bioloid.

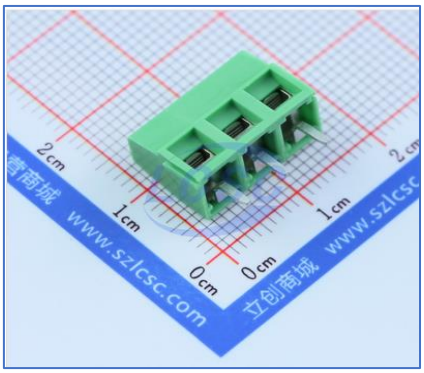


Ilustración 38: Bornera de 3 terminales.

- **Pines**: Se emplean para que la placa PCB o también denominada cape personalizada, pueda acoplar todos sus principales pines en los de la Beaglebone, haciendo que la placa quede unida a la Beagle.

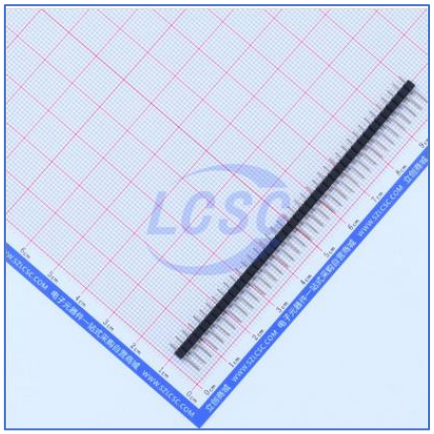


Ilustración 39: Pines macho de Conexión.

- **Unidad Inercial de Medida (IMU):** Se ha empleado el modelo MPU-9250 fabricado por TDK InvenSense. La elección de este componente se debe a que permite tener un magnetómetro, un giróscopo y un acelerómetro en un mismo dispositivo, sin tener que hacer variaciones en la comunicación por I2C.

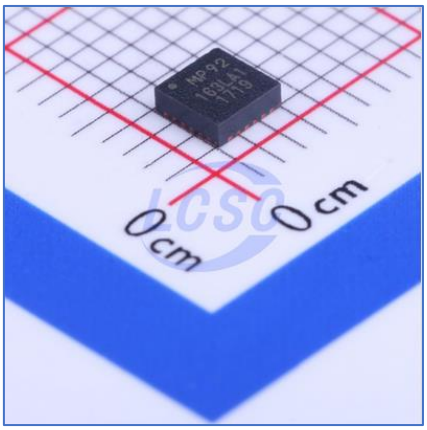


Ilustración 40: IMU MPU-9250

- **Batería porta pilas**: Dado que tanto la Beaglebone como el Robot Bioloid reciben energía mediante la batería del Robot, era necesario pensar en una fuente externa para los servos, de modo que si en caso contrario, estos se alimentasen mediante la Beagle, era probable que esta se resetease o que la batería se agotara antes. Por ello, la elección de una alimentación externa era crucial. Un punto importante era que, además, pudiera estar acoplada en la placa, por lo que la mejor opción era alimentar los servos mediante dos pilas de botón de 3V conectadas en serie. De ahí la elección de este componente.

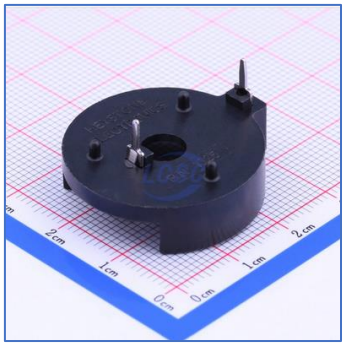


Ilustración 41: Portapilas.

- **Módulo Bluetooth**: Se ha escogido el módulo HC-05 dado que es un dispositivo sencillo de configurar y de comunicar.

Ilustración 42: Módulo HC-05

- **Pilas CR2430**: Este modelo de pila permite meter dos pilas en serie de 3V en el porta pilas seleccionado, por lo que cumplen con su objetivo de alimentar los servos.

Ilustración 43: Pila CR-2430 de 3V.

9.2.3. Diseño

A continuación, se explican las partes más importantes del diseño del circuito, para ello, este se divide en tres partes principales, que son las siguientes:

- Control Servomotores:

En esta parte se incluye la porta pilas y las dos borneras de los servos que cierran el circuito conectando el pin de masa a la masa de la Beagle. Añadir que los terminales 1 de cada bornera están relacionados con las señales PWM de la Beagle, por lo que para mejorar la forma de estas señales se le conecta una resistencia a la masa.

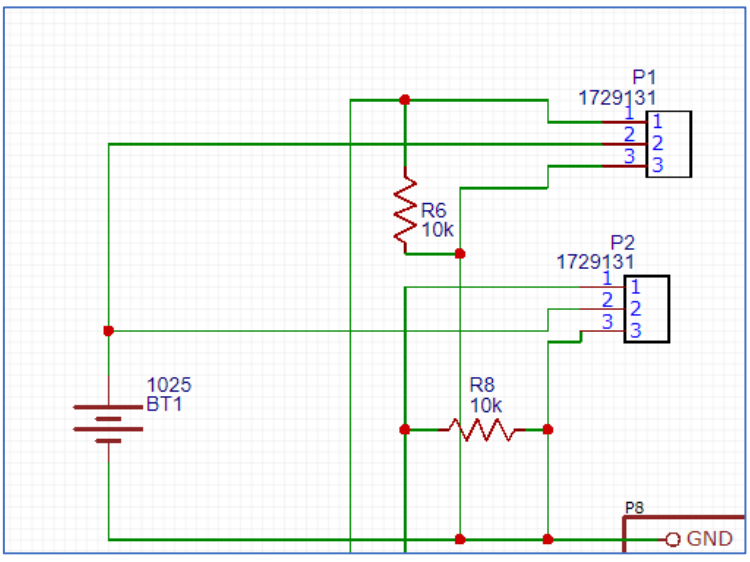


Ilustración 44: Circuito de los servomotores.

- Circuito IMU:

Esta parte incluye la alimentación del sensor en VDDIO y VDD, la dirección del dispositivo en I2C (AD0/SDO), la selección del protocolo de comunicación I2C

(nCS) y por supuesto, los pines de comunicación SDA y SCL. Del mismo modo que con las señales PWM, estas llevan una resistencia conectada, pero en modo Pull-Up.

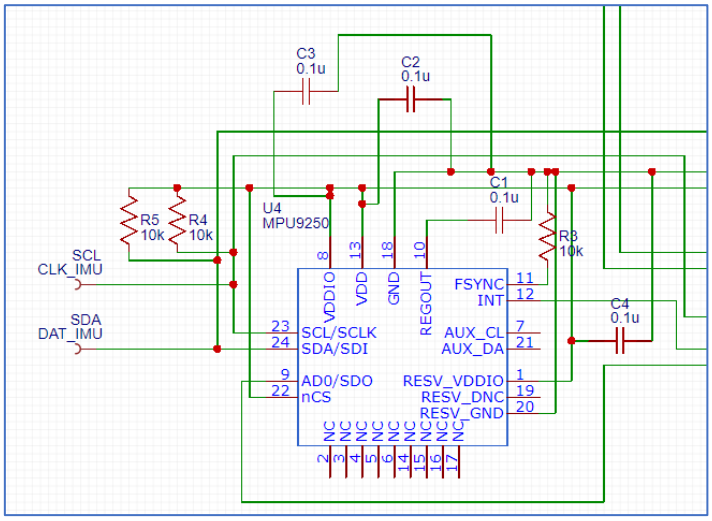


Ilustración 45: Circuito IMU.

- Circuito Bluetooth:

Otra parte de gran importancia se centra en el módulo Bluetooth donde se situa el componente HC-05 y este lleva conectados los LEDs explicados anteriormente aparte de los pines de alimentación y masa además de como es evidente, los pines de comunicación que en este caso son de protocolo UART. Para poder llevar a cabo una correcta utilización del módulo ha sido necesario dejar el pin correspondiente al PIO11 en abierto con un terminal para poder soldarle un cable que se conecte a Vcc y GND.

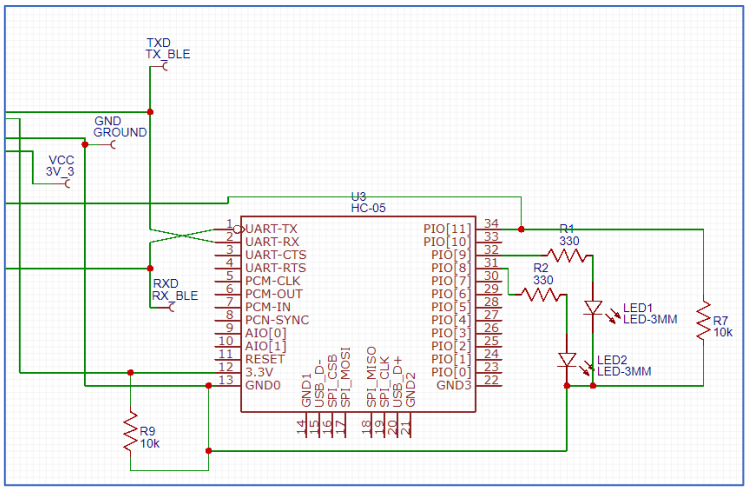
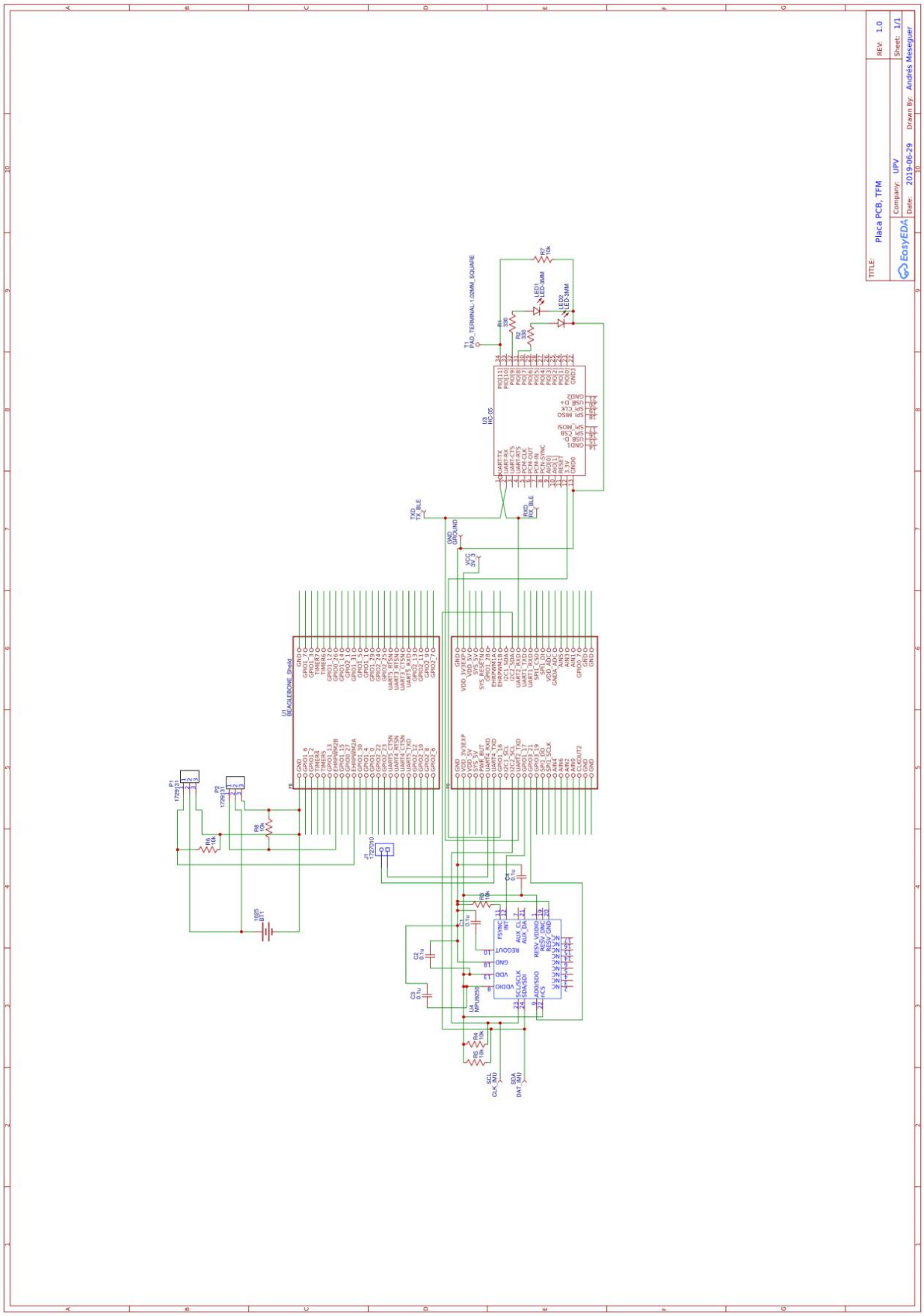


Ilustración 46: Circuito módulo Bluetooth.

A partir de aquí solo queda adjuntar todo el circuito completo para poder observar los pines exactos escogidos de la Beagle para transmitir la comunicación mediante I2C y UART, además del control de los servos mediante PWM.


TITLE: Placa PCB, TFM
REV: 1.0
Sheet: 1/1
EasyEDA
Company: UPV
Date: 2019-06-29
Drawn By: Andrés Meseguer

La cape personalizada se ha diseñado utilizando también la herramienta *EasyEDA*, ubicando los componentes integrados en el circuito diseñado en las posiciones que se requería. Pero el hecho de que varios componentes por sus dimensiones han creado diversos problemas de posicionamiento y funcionamiento, se han diseñado varias versiones.

La primera versión, aunque tiene problemas de diseño es la versión que funciona, en cambio la segunda versión ha sufrido problemas con la corriente y tensión que administran los puertos GPIO de la Beaglebone, aunque la posición de los componentes esté bastante mejor escogida.

9.2.4. Placa PCB v1.0

Los criterios principales de diseño han sido los siguiente:

- Asegurar el funcionamiento del módulo Bluetooth, dejando por lo tanto el pin PIO11 en abierto.
- Ubicar el módulo Bluetooth para evitar problemas de alcance de la señal.
- Encaminamiento realizado automáticamente con pistas de 0.1 mm de anchura.
- Se han puesto diferentes rotulaciones con el logotipo de la universidad, la versión de la placa, el nombre del autor y del tutor, además de los componentes y su polaridad.

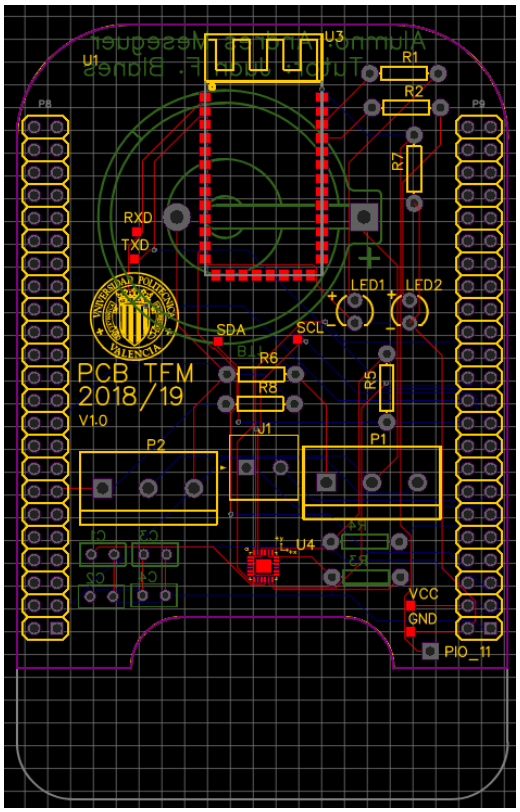


Ilustración 47: Diseño PCB v1.0 en EasyEDA.

Se han encontrado los siguientes defectos en el diseño:

- Poner la porta pilas en la cara inferior ha sido un error grave debido a que no hay espacio físico para poder acoplar la placa a la Beagle ya que el componente es demasiado alto. La solución ha sido ponerla en la parte superior, justo encima del módulo Bluetooth
- Los cables conectados a las borneras pueden provocar ruidos en las medidas obtenidas de la IMU.
- El pin PIO11 se ha dejado abierto, lo que obliga a la intervención humana para asegurar que el módulo Bluetooth funcione.

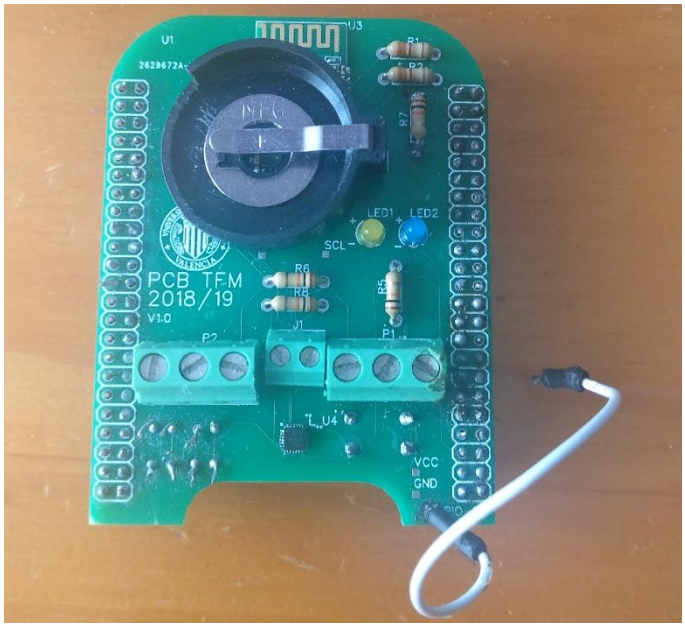


Ilustración 48: Placa v1.0 montada (Vista Frontal).

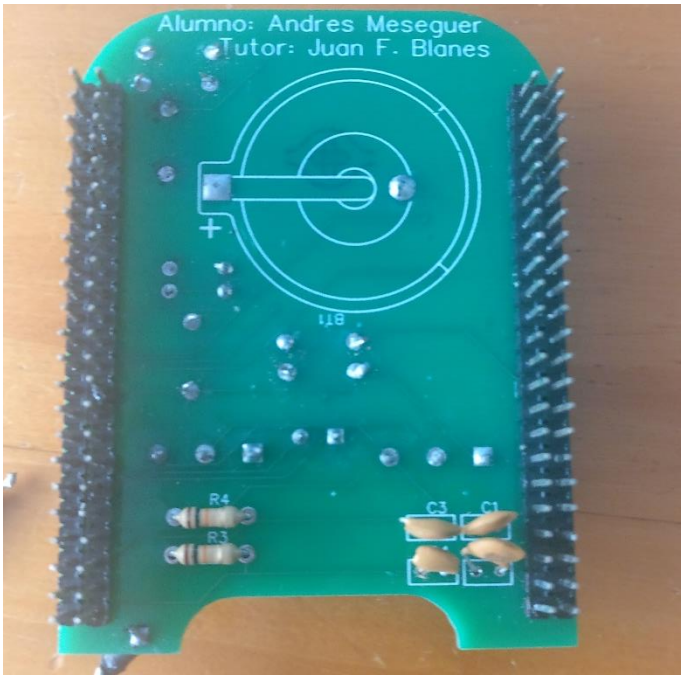


Ilustración 49: Placa v1.1 montada (Vista Trasera).

9.2.5. Placa PCB v1.1.

Los criterios principales de diseño han sido los siguiente:

- Se ha asegurado que las borneras estuvieran lo más cerca posible del margen inferior para evitar problemas con la longitud del cableado de los servos.
- La porta pilas ha sido necesario ubicarla en la parte superior debido a que tiene una altura excesiva para ubicarla en la parte trasera.
- La IMU se ha puesto de forma estratégica para que ninguno de los cables pasara por encima, evitando de la mejor forma posible, la perturbación en el campo magnético provocada por las pistas de la placa.
- Se han añadido diferentes rotulaciones indicando la polaridad de los LEDs, los nombres de los componentes, la función de cada entrada de las borneras además de información documental como puede ser el logotipo de la universidad, la versión de la placa, el proyecto y en la parte trasera, los nombres del autor y el tutor.
- El encaminamiento se ha hecho de forma automática de modo que los pines tuvieran la mayor anchura posible (0.19 mm) para evitar problemas con las intensidades, factibles ante todo en la parte dedicada a la alimentación de los servos.
- El módulo Bluetooth tiene el pin PIO11 y la alimentación, conectados a varios puertos GPIO para hacer que este componente funcione sin intervención humana.

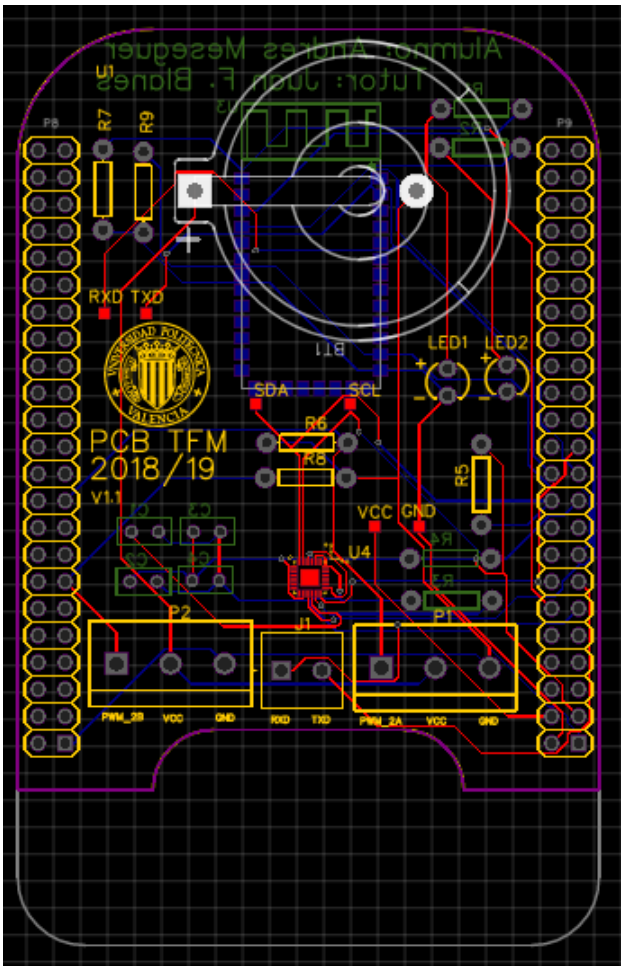


Ilustración 50: Diseño placa v1.1 en EasyEDA

Problemas

- El módulo Bluetooth no se puede accionar debido a que los puertos GPIO no ofrecen intensidad suficiente para alimentar el módulo.

Ilustración 51: PCB v1.1 montada, cara superior

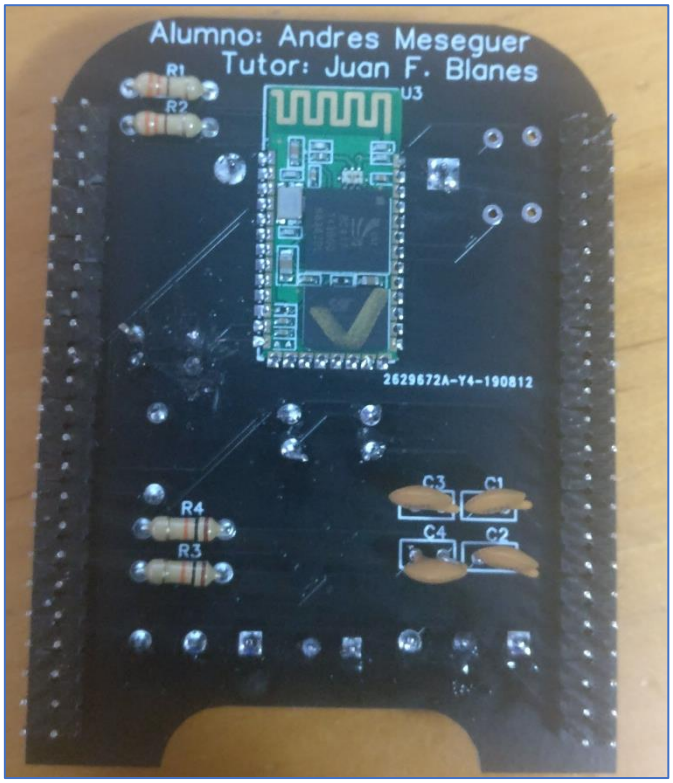


Ilustración 52: PCB v1.1 montada, cara inferior

De las dos versiones, finalmente se ha empleado la versión 1.0 debido a que es la que logra hacer funcionar todas las distintas partes.

9.3. Instalación de la Beagle

Es necesario seguir estas instrucciones:

- Insertar una tarjeta microsd de 8 gb mínimo de capacidad, en un ordenador Linux.
- Buscar la última versión disponible de la imagen de debian hard float a instalar en la Beaglebone. En este caso, se ha encontrado la versión 9.9 (15-06-2019). Para descargarla aplicar en consola: **wget** https://rcn-ee.com/rootfs/2019-06-15/microsd/bone-debian-9.9-console-armhf-2019-06-15-2gb.img.xz
- En el directorio descargado, aplicar el comando para descomprimir este archivo .xz: **xz -d bone-debian-9.9-console-armhf-2019-06-15-2gb.img.xz**
- Una vez descomprimido este archivo, se obtiene la imagen resultante, por lo tanto, ahora es momento de trabajar con la memoria SD. Para ello hay que distinguirla con el comando: **ls -l /dev/sd* .** Normalmente aparece como el dispositivo sdb, pero para asegurarse es conveniente quitarla y aplicar el mismo comando para comparar resultados y cercenarse de que efectivamente, se trata del dispositivo sdb.
- El siguiente paso es desmontar las particiones que aparezcan junto son sdb. En este caso sdb1. Comando: **umount /dev/sdb1**
- A partir de aquí ya es posible volcar la imagen sobre la memoria al aplicar: **sudo dd if=bone-debian-9.9-console-armhf-2019-06-15-2gb.img of=/dev/sdb bs=1M**
- Con la sd ya montada, el siguiente paso es trabajar en la Beagle. La forma más sencilla es conectando el dispositivo al puerto USB del ordenador mediante un cable miniUSB a USB. Recordar la previa introducción de la memoria sd montada en la Beagle.
- Se emplea el comando: **ssh** debian@192.168.7.2 para establecer una conexión mediante ssh, la cual, para establecerse, pide una contraseña relacionada con el usuario **debian**. Dicha contraseña es predefinidamente: **temppwd**
- **Una vez dentro, es importante asegurarse que la beagle tenga acceso a internet, para ello se aplica el comando apt update siendo el root del**

dispositivo. En caso de tener problemas de conexión, lo más plausible es que la beagle no tenga acceso a la red, por lo tanto, hay que seguir los pasos descritos en la siguiente página web: http://jpdelacroix.com/tutorials/sharing-internet-beaglebone-black.html

- Tras estos pasos, el usuario se sitúa dentro de la Beagle con la imagen montada. Por lo tanto, es necesario asegurarse de que el dispositivo haya sido correctamente instalado, para ello aplicar el comando: **uname –a**. La salida de este comando por ejemplo puede ser: Linux arm 4.14.108-ti-r107 #1 SMP PREEMPT Wed Jun 12 05:21:36 UTC 2019 armv7l GNU/Linux
- Posteriormente se actualiza la versión de firmware para poder introducir herramientas de tiempo real. Comando: **/opt/scripts/tools/update_kernel.sh --bone-rt-kernel --lts-4_14**
- El siguiente paso es expandir la partición que se está empleando para trabajar con la Beagle dentro de la memoria sd. Para ello se reinicia la Beagle tras actualizar el kernel y se introducen los siguientes comandos:

```
root@arm:~# ls -l /dev/mmcblk*
brw-rw---- 1 root disk 179,  0 Jun 16 03:31 /dev/mmcblk0
brw-rw---- 1 root disk 179,  1 Jun 16 03:31 /dev/mmcblk0p1
brw-rw---- 1 root disk 179,  8 Jun 16 03:31 /dev/mmcblk1
brw-rw---- 1 root disk 179, 16 Jun 16 03:31 /dev/mmcblk1boot0
brw-rw---- 1 root disk 179, 24 Jun 16 03:31 /dev/mmcblk1boot1
brw-rw---- 1 root disk 179,  9 Jun 16 03:31 /dev/mmcblk1p1
brw-rw---- 1 root disk 179, 32 Jun 16 03:31 /dev/mmcblk1rpmb

root@arm:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            216M     0  216M   0% /dev
tmpfs            49M  6.0M   43M  13% /run
/dev/mmcblk0p1  1.6G  1.2G  406M  74% /
tmpfs           242M     0  242M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           242M     0  242M   0% /sys/fs/cgroup
tmpfs            49M     0   49M   0% /run/user/1000
```

Sabiendo que la memoria es de 32 GB, es posible entender que solo se usan 1.6 GB de 32 posibles, por lo que es interesante expandir la partición:

```
root@arm:/home/debian# /opt/scripts/tools/grow_partition.sh
Salida indiferente
root@arm:/home/debian# df -h
Filesystem      Size  Used Avail Use% Mounted on
```

```
udev          220M     0  220M   0% /dev
tmpfs          50M  5.6M   44M  12% /run
/dev/mmcblk0p1   30G  1.2G   27G   4% /
tmpfs         246M     0  246M   0% /dev/shm
tmpfs         5.0M  4.0K  5.0M   1% /run/lock
tmpfs         246M     0  246M   0% /sys/fs/cgroup
tmpfs          50M     0   50M   0% /run/user/1000
```

Con el resultado del último comando es posible apreciar que se ha expandido correctamente la memoria.

- Posteriormente hay que instalar el driver de la cámara uEye empleada. [9]

Accediendo a la Beagle como usuario root, realizar los siguientes pasos:
root@arm:/home/debian# apt update
root@arm:/home/debian# apt upgrade
Descargar el archivo con todo lo necesario para instalar los drivers de la cámara:
root@arm:/home/debian# wget https://en.ids-imaging.com/download-ueye-emb-hardfloat.html?file=tl_files/downloads/LinuxEmbedded/uEyeSDK-4.92.00-ARM_LINUX_IDS_GNUEABI_HF.tgz
Descomprimir el archivo:
root@arm:/home/debian# tar xvf download-ueye-emb-hardfloat.html?file=tl_files%2Fdownloads%2FLinuxEmbedded%2FuEyeSDK-4.92.00-ARM_LINUX_IDS_GNUEABI_HF.tgz
Tras descomprimir, acceder al directorio extraido del archivo y ejecutar el .run ubicado en esta carpeta para que se instalen los ficheros relacionados con la cámara.
root@arm:/home/debian# cd build_linux_armhf_raspbian_ids
root@arm:/home/debian/build_linux_armhf_raspbian_ids# ls
ueye_4.92.0.0_armhf.run
root@arm:/home/debian/build_linux_armhf_raspbian_ids# ./ueye_4.92.0.0_armhf.run
ueye 4.92.0.0, Copyright (c) IDS Imaging Development Systems GmbH

This is a self-extracting archive.
The archive will be extracted to: /

If you want to stop extracting, please press <ctrl-C>.

Do you want to continue? [Y/n]:
Y

Using target directory: /
Extracting, please wait...

Unpacking finished successfully
Info: Group 'ueye' created.
Info: User 'ueyed' created.
Info: Config /etc/ids/ueye/ueyeusbd.conf created.
Info: Config /etc/ids/ueye/ueyeapimachine.conf created.
Info: Run directory '/run/ueyed' created.
Info: Autostart for USB driver enabled.

```
Info: Config /etc/ids/ueye/ueyeethd.conf created.
Info: Run directory '/run/ueyed' created.
Info: Autostart for ETH driver enabled.
```

- Una vez instalado el driver de la cámara, será necesario instalar los componentes adicionales básicos para poder usar el compilador en el sistema empotrado junto con las librerías de la cámara y la versión de opencv a instalar posteriormente.

```
root@arm:/usr/lib# sudo apt update
root@arm:/usr/lib# sudo apt upgrade
root@arm:/usr/lib# sudo dpkg --add-architecture armhf
root@arm:/usr/lib# sudo apt update
root@arm:/usr/lib# sudo apt install qemu-user-static
root@arm:/usr/lib# sudo apt-get install python3-dev
root@arm:/usr/lib# sudo apt-get install python3-numpy
root@arm:/usr/lib# sudo apt-get install python-dev
root@arm:/usr/lib# sudo apt-get install python-numpy
root@arm:/usr/lib# sudo apt-get install libpython3-dev:armhf

root@arm:/usr/lib# sudo apt install libgtk-3-dev:armhf libcanberra-gtk3-dev:armhf
root@arm:/usr/lib# sudo apt install libtiff-dev:armhf zlib1g-dev:armhf
root@arm:/usr/lib# sudo apt install libjpeg-dev:armhf libpng-dev:armhf
root@arm:/usr/lib# sudo apt install libavcodec-dev:armhf libavformat-dev:armhf libswscale-
dev:armhf libv4l-dev:armhf

root@arm:/usr/lib# sudo apt-get install libxvidcore-dev:armhf libx264-dev:armhf
root@arm:/usr/lib# sudo apt install cmake git pkg-config wget
```

- Instalación de OpenCV 4. [10]

```
root@arm:/usr/lib# cd ~
root@arm:~# mkdir opencv_all && cd opencv_all
root@arm:~/opencv_all# wget -O opencv.tar.gz
https://github.com/opencv/opencv/archive/4.1.0.tar.gz
root@arm:~/opencv_all# tar xf opencv.tar.gz
root@arm:~/opencv_all# wget -O opencv_contrib.tar.gz
https://github.com/opencv/opencv_contrib/archive/4.1.0.tar.gz
root@arm:~/opencv_all# tar xf opencv_contrib.tar.gz
root@arm:~/opencv_all# rm *.tar.gz
root@arm:~/opencv_all# export PKG_CONFIG_PATH=/usr/lib/arm-linux-
gnueabihf/pkgconfig:/usr/share/pkgconfig
root@arm:~/opencv_all# export PKG_CONFIG_LIBDIR=/usr/lib/arm-linux-
gnueabihf/pkgconfig:/usr/share/pkgconfig
root@arm:~/opencv_all# cd opencv-4.1.0
root@arm:~/opencv_all/opencv-4.1.0# mkdir build && cd build
```

```
root@arm:~/opencv_all/opencv-4.1.0/build# cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/opt/opencv-4.1.0 \
    -D CMAKE_TOOLCHAIN_FILE=../platforms/linux/arm-gnueabi.toolchain.cmake \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_all/opencv_contrib-4.1.0/modules \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D BUILD_TESTS=OFF \
    -D BUILD_DOCS=OFF ..
 root@arm:~/opencv_all/opencv-4.1.0/build#     make -j $(($(nproc) + 1))
```

9.4. Código Final

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include "HC_05.hpp"
#include "ueye.h"
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>
#include "opencv2/objdetect.hpp"
#include "UART.hpp"
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include "PWMuniv.h"


#define       DEVICE_PORT_1          "/dev/ttyO4"

using namespace cv;
using namespace std;

HIDS hCam = 0;
char Values[6];

char* pMem = NULL;
UINT memID = 0;
int capturasNegativas =-1;
int captura =0;

char nombre[256];

char buff[128];



struct Mensaje{
      char LowByte;
```

```
	char HighByte;
	char *Values;
	bool Comunicacion;
};

void MoverSoporte();

void Inicializar(struct Mensaje *MSG){
	MSG->Values=(char *)malloc(6 * sizeof(char));
	MSG->Values[0]=(char)0xff;
	MSG->Values[1]=(char)0x55;
	MSG->Values[2]=(char)0xdc;
	MSG->Values[3]=(char)(~0xdc);
	MSG->Values[4]=(char)0xab;
	MSG->Values[5]=(char)(~0xab);
	MSG->Comunicacion=false;

	Values[0]=(char)0xff;
	Values[1]=(char)0x55;
	Values[2]=(char)0xdc;
	Values[3]=(char)(~0xdc);
	Values[4]=(char)0xab;
	Values[5]=(char)(~0xab);


}




bool Decodificar(char msg[128], char Low, char High){
	char ret[2];
	bool reconocido = false;
	for(volatile int i=0;i<128;i++){
		if(msg[i]== 0xff && msg[i+1]==0x55){
				ret[0]=msg[i+4];
				ret[1]=msg[i+2];
				reconocido=true;
				break;
		}
	}
	if(reconocido){
		if(ret[1]==Low && ret[0]==High)
			return true;
	}else{
		return false;
	}
}


char * ObtenerQRData(){
```

```
   cv::Mat inputImage = cv::imread("captura.png", 1); //Carga de la imagen guar
dada
   if(!inputImage.data) {
      is_FreeImageMem (hCam,pMem,memID);
            is_ExitCamera(hCam);
      std::cout << "Unable to open the image file" << std::endl;
      return 0;
   }
   cv::Mat bwsrc;

   cvtColor(inputImage, bwsrc, cv::COLOR_RGB2GRAY); //Conversión a grises

      QRCodeDetector qrDecoder = QRCodeDetector();
      Mat bbox, rectifiedImage;
      //Detección y decodificación
      std::string data = qrDecoder.detectAndDecode(bwsrc, bbox, rectifiedImage
);
      if(data.length()>0)
      {
       if(data.length()==9){ //9
            Values[2]=(char)0x04;
            Values[3]=(char)(~0x04);
            Values[4]=(char)0x09;
            Values[5]=(char)~(0x09);
            capturasNegativas=0;

            cout << "Enviando Izquierda "<<endl;

       }else if(data.length() == 7){ //Derecha
            Values[2]=(char)0x02;
            Values[3]=(char)~(0x02);
            Values[4]=(char)0x05;
            Values[5]=(char)~(0x05);
            capturasNegativas=0;

            cout << "Enviando Derecha "<<endl;

       }else if(data.length() == 5){ //5 Recto
            Values[2]=(char)0x08;
            Values[3]=(char)~(0x08);
            Values[4]=(char)0x03;
            Values[5]=(char)~(0x03);
            cout << "Enviando Recto "<<endl;
            capturasNegativas=0;

       }else{
            Values[2]=(char)0x05;
            Values[3]=(char)~(0x05);
            Values[4]=(char)0x00;
            Values[5]=(char)~(0x00);
            cout <<"Codigo valido no entendible"<<endl;
            capturasNegativas++;
```

```
        }
      }
      else{
        cout << "QR Code not detected" << endl;
        Values[2]=(char)0x05;
        Values[3]=(char)~(0x05);
        Values[4]=(char)0x00;
        Values[5]=(char)~(0x00);
        capturasNegativas++;
      }
      Values[0]=(char)0xff;
      Values[1]=(char)0x55;
}

void *Capturadora(void *message){
      struct Mensaje *MSG= (struct Mensaje*)message;
      int nRet; //Almacena los retornos de las funciones del uEye
      //Variables locales
      UINT count, val_pclock = 9;//No poner a 30 pq en la Beagle no funciona
      UINT bytesNeeded = sizeof(IMAGE_FORMAT_LIST);

      nRet = is_InitCamera(&hCam, NULL); //Inicia el driver y la comunicación
con la cámara

      if(nRet != IS_SUCCESS){ //Si no se ha iniciado correctamente, sal del pr
ograma
            cout << "Error al iniciar la camara. " << endl;
            cout << nRet << endl;
            return 0;
      }
      else{cout << "Camara Iniciada." << endl;}
            //Ajuste de anchura de los píxels de entrada
            nRet = is_PixelClock(hCam, IS_PIXELCLOCK_CMD_SET,(void*)&val_pcl
ock, sizeof(val_pclock));

            if(nRet == IS_SUCCESS) cout << "Exito al modificar el valor de p
ixelclock" << endl;
            else cout << "Fallo al modificar el valor de pixelclock. Error n
o: " <<nRet << endl;

            //Se consigue una lista de los formatos posibles
            nRet = is_ImageFormat(hCam, IMGFRMT_CMD_GET_NUM_ENTRIES, &count,
4);
            bytesNeeded += (count - 1) * sizeof(IMAGE_FORMAT_INFO);
            void* ptr = malloc(bytesNeeded);
            IMAGE_FORMAT_LIST* pformatList = (IMAGE_FORMAT_LIST*) ptr;
            pformatList->nSizeOfListEntry = sizeof(IMAGE_FORMAT_INFO);
            pformatList->nNumListElements = count;
            nRet = is_ImageFormat(hCam, IMGFRMT_CMD_GET_LIST, pformatList, b
ytesNeeded);
```

```
        nRet = is_PixelClock(hCam,IS_PIXELCLOCK_CMD_GET,(void*)&val_pclo
ck,sizeof(val_pclock));
        cout << "El valor actual del pixelclock es: " << val_pclock;
        cout << endl << " Comenzando captura..." << endl << endl;
        IMAGE_FORMAT_INFO formatInfo;
        IMAGE_FILE_PARAMS file_param;
        //Especificación del nombre de la imagen
        sprintf(nombre, "./captura.png",captura);

        const size_t tam = strlen(nombre)+1;
        wchar_t* file_name = new wchar_t[tam];
        mbstowcs (file_name, nombre, tam);

        //Propiedades del archivo
        file_param.pwchFileName = file_name;
        file_param.pnImageID = NULL;
        file_param.ppcImageMem = NULL;
        file_param.nQuality = 100; //Mejora un poco la nitidez
        file_param.nFileType = IS_IMG_PNG;
        //Establecimiento de anchura y altura según el formato de imagen
        formatInfo = pformatList->FormatInfo[9];

        int width = formatInfo.nWidth;
        int height = formatInfo.nHeight;
        cout << formatInfo.nWidth << "x" << formatInfo.nHeight << endl;
        //Bucle de captura...
        while (nRet == IS_SUCCESS){

            nRet = is_AllocImageMem(hCam, width, height, 24, &pMem,(
int*) &memID);
            if (nRet != IS_SUCCESS){
                break;
            }

            //Activación de la memoria
            nRet = is_SetImageMem(hCam, pMem, memID);
            //Se especifica el formato de la nueva imagen
            nRet = is_ImageFormat(hCam, IMGFRMT_CMD_SET_FORMAT, &for
matInfo.nFormatID, 4);
            //Adquisición de la imagen
            nRet = is_FreezeVideo(hCam, IS_WAIT);


            //Guardado de la imagen
            nRet = is_ImageFile(hCam, IS_IMAGE_FILE_CMD_SAVE, (void*
)&file_param, sizeof(file_param));
            if(nRet != 0){
                is_FreeImageMem (hCam,pMem,memID);
                is_ExitCamera(hCam);
                break;
            }
            MSG->Comunicacion=false;
```

```
            MSG->Values=ObtenerQRData(); //Función obtención del cód
            MoverSoporte();
            MSG->Comunicacion=true;
            usleep(3E6);

        }


        //Liberación de la cámara
        is_FreeImageMem (hCam,pMem,memID);
        is_ExitCamera(hCam);
        free(ptr);
}


void *ComunicacionBioloid(void *message){
    struct Mensaje *MSG= (struct Mensaje*)message;
    for(volatile int i=0;i++;i<6){
        Values[i]=MSG->Values[i];
    }

    string command_pin_RX = "/usr/bin/config-pin P9.11 uart";
    string command_pin_TX = "/usr/bin/config-pin P9.13 uart";
    system(command_pin_RX.c_str());
    system(command_pin_TX.c_str());
    BBB::UART  myUart(BBB::UART4,
                      BBB::Baud57600,
                      BBB::ParityNo,
                      BBB::StopOne,
                      BBB::Char8 );

    myUart.open( BBB::ReadWrite );
    myUart.flush( BBB::bothDirection );
    while(!Decodificar(buff,(char)0xA0,(char)0x80)){
        memset(buff, 0, sizeof buff);
        myUart.write(Values, sizeof(Values));
        usleep(1000000);
        myUart.read(buff, sizeof(buff));
        cout<<buff<<endl;
    }
  myUart.read(buff, sizeof(buff));
  myUart.read(buff, sizeof(buff));

    MSG->Comunicacion=true;
    myUart.close();

    usleep(1000);
    myUart.open( BBB::ReadWrite );
    myUart.flush( BBB::bothDirection );


    cout<<"Entrada bucle principal Comunicacion Bioloid"<<endl;
```

```
	char *mensaje=(char *)malloc(6 * sizeof(char));;
	while(1){

		//Bucle enviar orden
			memset(buff, 0, sizeof buff);
			while(!MSG->Comunicacion){}
			myUart.write(Values, sizeof(Values));
			usleep(1000000);
			myUart.read(buff, sizeof(buff));
	}
	myUart.close();

}
void *Ble_Imu(void *message){
	struct Mensaje *MSG= (struct Mensaje*)message;
	using namespace HC;
	using namespace BBB;
	string x;
	HC_05 ble;
	usleep(1E6);
	ble.Enviar(FUZZY);
   usleep(1E6);
   ble.Enviar(GET_COM_STATUS);
	usleep(1E6);
	ble.Enviar(SET_NAME);
	usleep(1E6);
	ble.Enviar(SET_ROLE_SLAVE);
	usleep(1E6);
	ble.Enviar(GET_STATE);
	usleep(1E6);
	ble.Enviar(GET_ADDRESS);
	usleep(1E6);
	ble.Enviar(GET_VERSION);
	usleep(1E6);
	ble.Enviar(BAUD_RATE);
	usleep(1E6);
	ble.Enviar(RESET);
	usleep(1E6);

	ble.Transmitir();
}

void MoverSoporte(){
	using namespace BBB;
	int fase= capturasNegativas;
	PWM pwm(PWM::P8_13);
	pwm.setPolarity(PWM::ACTIVE_HIGH);
	pwm.setPeriod(20E6);

	PWM pwm2(PWM::P8_19);
	pwm2.setPolarity(PWM::ACTIVE_HIGH);
	pwm2.setPeriod(20E6);
```

```
float x=8.00;
float duty = 8.0;
switch(fase){
    case 0:
        pwm.run();
        for (int i=0; i<10; i++) {//al centro
                pwm.setDutyCycle((float)8.0);
                usleep(2000);
        }
        pwm.stop();
        pwm2.run();
          for (int i=0; i<10; i++) {//De delante a enmedio
                    pwm2.setDutyCycle((float)8.0);
                    usleep(2000);
            }
            pwm2.stop();
        break;
    case 4:
        pwm.run();
        for (int i=0; i<52; i++) {//De centro a derecha
                x -=(0.05);
                    pwm.setDutyCycle((float)x);
                    usleep(20000);
            }
        pwm.stop();
        pwm2.run();
        for (int i=0; i<10; i++) {//De enmedio hacia delante
                duty -=0.1;
                pwm2.setDutyCycle((float)duty);
                usleep(200000);

        }
        pwm2.stop();
        break;
    case 5:
        pwm.run();
        //De derecha a izquierda
        for (int i=0; i<65; i++) {
            x +=(0.05);
                pwm.setDutyCycle((float)x);
                usleep(20000);
        }
            pwm.stop();
        break;
    case 6:
        pwm.run();
            //De izquierda al centro
        for (int i=0; i<30; i++) {
                x -=(0.05);
                pwm.setDutyCycle((float)x);
                usleep(20000);
        }
```

```
                pwm.stop();
                //De delante a enmedio
                pwm2.run();
            for (int i=0; i<10; i++) {
                    duty +=0.1;
                        pwm2.setDutyCycle((float)duty);
                        usleep(20000);
                }
                pwm2.stop();
                capturasNegativas=-1;
                break;
        default:

                break;
    }
}

int main() {
    string str = "/etc/init.d/ueyeusbdrc start";
    const char *command = str.c_str();
    system(command);

    //Para que conecte con la primera cámara que encuentre
    struct Mensaje MSG;
    pthread_attr_t atrib; //Atributo de los hilos
    pthread_t   Bioloid, Camara, BLE;
    pthread_attr_init(&atrib); //Incialización de atributos
    Inicializar(&MSG);
    pthread_create(&BLE,&atrib,Ble_Imu,(void*)&MSG); //Creación del hilo bl$
    usleep(10E6);

    pthread_create(&Bioloid,&atrib,ComunicacionBioloid,(void*)&MSG); //Creac
ión del hilo monitor
    while(MSG.Comunicacion==false){
        usleep(10E4);
    }

    pthread_create(&Camara,&atrib,Capturadora,(void*)&MSG); //Creación del h
ilo monitor

    cout<<"Hilos Iniciados"<<endl;
    pthread_join(Camara,NULL);
    pthread_join(BLE,NULL);

}
```

9.5. Código Aplicación Escritorio

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using InTheHand;
using InTheHand.Net.Bluetooth;
using InTheHand.Net.Ports;
using InTheHand.Net.Sockets;
using System.IO;
using System.Globalization;
using System.Windows.Forms.DataVisualization.Charting;

namespace Bluetooth_Tutorial
{
   public partial class Form1 : Form
   {
      delegate void SetChartCallback();

      float MagX, MagY, MagZ, Mod_Mag;


      float AccX, AccY, AccZ, Mod_Acc;

      float GyrX, GyrY, GyrZ, Mod_Gyr;


      string Time, direccion;
      int it = 0;

      List<string> items;
      public Form1()
      {
         items = new List<string>();
         InitializeComponent();

         chart1.Titles.Add("Campo Magnético");
         chart1.Titles.Add("Modulo = ¿?");


         chart1.ChartAreas[0].AxisX.Title = "Tiempo";
         chart1.ChartAreas[0].AxisY.Title = "Magnetismo (uT)";

         chart1.Series.Add("MagX");
         chart1.Series["MagX"].BorderWidth = 6;
         chart1.Series["MagX"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

         chart1.Series.Add("MagY");
         chart1.Series["MagY"].BorderWidth = 6;
         chart1.Series["MagY"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

         chart1.Series.Add("MagZ");
         chart1.Series["MagZ"].BorderWidth = 6;
         chart1.Series["MagZ"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;
```

```
        chart1.Series["MagX"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart1.Series["MagY"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart1.Series["MagZ"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;


        //chart2.Series.Clear();
        chart2.Titles.Add("Aceleración");
        chart2.Titles.Add("Modulo = ¿?");


        chart2.ChartAreas[0].AxisX.Title = "Tiempo";
        chart2.ChartAreas[0].AxisY.Title = "Aceleración (g)";

        chart2.Series.Add("AccX");
        chart2.Series["AccX"].BorderWidth = 6;
        chart2.Series["AccX"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

        chart2.Series.Add("AccY");
        chart2.Series["AccY"].BorderWidth = 6;
        chart2.Series["AccY"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

        chart2.Series.Add("AccZ");
        chart2.Series["AccZ"].BorderWidth = 6;
        chart2.Series["AccZ"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

        chart2.Series["AccX"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart2.Series["AccY"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart2.Series["AccZ"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;



        //
        chart3.Titles.Add("Giroscopio");
        chart3.Titles.Add("Modulo = ¿?");

        chart3.ChartAreas[0].AxisX.Title = "Tiempo";
        chart3.ChartAreas[0].AxisY.Title = "Gircoscopio (º/s)";

        chart3.Series.Add("GyrX");
        chart3.Series["GyrX"].BorderWidth = 6;
        chart3.Series["GyrX"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

        chart3.Series.Add("GyrY");
        chart3.Series["GyrY"].BorderWidth = 6;
        chart3.Series["GyrY"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;

        chart3.Series.Add("GyrZ");
        chart3.Series["GyrZ"].BorderWidth = 6;
        chart3.Series["GyrZ"].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String;
```

```
        chart3.Series["GyrX"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart3.Series["GyrY"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart3.Series["GyrZ"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;



    }

    private void bGo_Click(object sender, EventArgs e)
    {
      if (serverStarted)
      {
        updateUI("Server already started silly sausage!");
        return;
      }
      if (rbClient.Checked)
      {
        startScan();
        while (folderBrowserDialog1.ShowDialog() == DialogResult.Cancel)
        {


        }
        direccion = folderBrowserDialog1.SelectedPath;
        updateUI("@" + direccion + "\\Resultados.txt");
      }
      else
      {
        connectAsServer();
      }
    }


    private void startScan()
    {
      listBox1.DataSource = null;
      listBox1.Items.Clear();
      items.Clear();
      Thread bluetoothScanThread = new Thread(new ThreadStart(scan));
      bluetoothScanThread.Start();
    }
    BluetoothDeviceInfo[] devices;
    private void scan()
    {

      updateUI("Starting Scan..");
      BluetoothClient client = new BluetoothClient();
      devices = client.DiscoverDevicesInRange();
      updateUI("Scan complete");
      updateUI(devices.Length.ToString()+" devices discovered");
      foreach (BluetoothDeviceInfo d in devices)
      {
        items.Add(d.DeviceName);
      }

      updateDeviceList();
    }

    private void connectAsServer()
    {
```

```
        Thread bluetoothServerThread = new Thread(new ThreadStart(ServerConnectThread));
        bluetoothServerThread.Start();
    }

    private void connectAsClient()
    {
        throw new NotImplementedException();
    }


    Guid mUUID = BluetoothService.SerialPort;

bool serverStarted = false;
    public void ServerConnectThread()
    {
        serverStarted = true;
        updateUI("Server started, waiting for clients");
        BluetoothListener blueListener = new BluetoothListener(mUUID);
        blueListener.Start();
        BluetoothClient conn = blueListener.AcceptBluetoothClient();
        updateUI("Client has connected");

        Stream mStream = conn.GetStream();
        while (true)
        {
            try
            {
                //handle server connection
                byte[] received = new byte[2048];
                mStream.Read(received, 0, received.Length);
                updateUI("Received: " + Encoding.ASCII.GetString(received));
                byte[] sent = Encoding.ASCII.GetBytes("Hello World");
                mStream.Write(sent, 0, sent.Length);
            }
            catch (IOException exception)
            {
                updateUI("Client has disconnected!!!!");
            }
        }

    }

    private void updateUI(string message)
    {
        Func<int> del = delegate()
        {
            tbOutput.AppendText(message + System.Environment.NewLine);
            return 0;
        };
        Invoke(del);
    }

    private void updateDeviceList()
    {
        Func<int> del = delegate()
        {
            listBox1.DataSource = items;
            return 0;
        };
        Invoke(del);
    }
```

```
BluetoothDeviceInfo deviceInfo;
private void listBox1_DoubleClick(object sender, EventArgs e)
{
   deviceInfo = devices.ElementAt(listBox1.SelectedIndex);
   updateUI(deviceInfo.DeviceName + " was selected, attempting connect");

   if (pairDevice())
   {
      updateUI("device paired..");
      updateUI("starting connect thread");
      Thread bluetoothClientThread = new Thread(new ThreadStart(ClientConnectThread));
      bluetoothClientThread.Start();

   }
   else
   {
      updateUI("Pair failed");
   }

}

private void ClientConnectThread()
{
   BluetoothClient client = new BluetoothClient();
   updateUI("attempting connect");
   client.BeginConnect(deviceInfo.DeviceAddress, mUUID, this.BluetoothClientConnectCallback, client);

}

bool editado = false;

void BluetoothClientConnectCallback(IAsyncResult result)
{
   BluetoothClient client = (BluetoothClient)result.AsyncState;
   client.EndConnect(result);

   Stream stream = client.GetStream();
   stream.ReadTimeout = 1000;
   byte[] received = new byte[2048];
   string path = @direccion+"\\Resultados.txt";



   // Check if file already exists. If yes, delete it.

   while (true)
      {
         Thread.Sleep(1000);
         try
         {
            stream.Read(received, 0, received.Length);
            String msg = Encoding.ASCII.GetString(received);
            if (msg.Contains("Inicio") && msg.Contains("Fin")) {

               MagX = float.Parse(msg.Split(';')[1]);
               MagY = float.Parse(msg.Split(';')[2]);
               MagZ = float.Parse(msg.Split(';')[3]);
               Mod_Mag = float.Parse(msg.Split(';')[4]);
               AccX = float.Parse(msg.Split(';')[5]);
               AccY = float.Parse(msg.Split(';')[6]);
               AccZ = float.Parse(msg.Split(';')[7]);
               Mod_Acc= float.Parse(msg.Split(';')[8]);
               GyrX = float.Parse(msg.Split(';')[9]);
```

```
                    GyrY = float.Parse(msg.Split(';')[10]);
                    GyrZ = float.Parse(msg.Split(';')[11]);
                    Mod_Gyr = float.Parse(msg.Split(';')[12]);
                    Time = DateTime.Now.ToString("HH:mm:ss");

                this.ActualizarChart();

                // Add some text to file

                string txt = Time + ";" + MagX.ToString() + ";" + MagY.ToString() + ";" + MagZ.ToString() + ";" +
Mod_Mag.ToString() + ";" + AccX.ToString() + ";" + AccY.ToString() + ";" + AccZ.ToString() + ";" +
Mod_Acc.ToString() + ";"
                        + GyrX.ToString()+";"+GyrY.ToString()+";"+GyrZ.ToString()+";"+Mod_Gyr.ToString()+";" ;
                it=it+1;


                    using (var tw = new StreamWriter(path, true))
                    {
                      if (!editado)
                      {

tw.WriteLine("Hora;MagX;MagY;MagZ;Mod_Mag;AccX;AccY;AccZ;Mod_Acc;GirX;GirY;GirZ;Mod_Gir");
                          editado = true;
                      }
                    tw.WriteLine(txt);
                    tw.Close();
                    }

            }

            updateUI("Received: " + Encoding.ASCII.GetString(received));
            }
            catch { }
        }


    }
    int borrar = 0;

    private void ActualizarChart()
    {
      if (this.chart1.InvokeRequired)
      {
        SetChartCallback d = new SetChartCallback(ActualizarChart);
        this.Invoke(d, new object[] {  });
      }
      else
      {
        chart1.Series["MagX"].Points.AddXY(Time, MagX);
        chart1.Series["MagY"].Points.AddXY(Time, MagY);
        chart1.Series["MagZ"].Points.AddXY(Time, MagZ);
        chart1.Titles[1].Text = "Modulo = " + Mod_Mag.ToString() +" uT";
        if (it >= 50) {
          chart1.Series["MagX"].Points.RemoveAt(0);
          chart1.Series["MagY"].Points.RemoveAt(0);
          chart1.Series["MagZ"].Points.RemoveAt(0);
        }


      }

      if (this.chart2.InvokeRequired)
      {
```

```
            SetChartCallback d = new SetChartCallback(ActualizarChart);
            this.Invoke(d, new object[] { });
        }
        else
        {
            chart2.Series["AccX"].Points.AddXY(Time, AccX);
            chart2.Series["AccY"].Points.AddXY(Time, AccY);
            chart2.Series["AccZ"].Points.AddXY(Time, AccZ);
            chart2.Titles[1].Text = "Modulo = " + Mod_Acc.ToString()+" g";
            if (it >= 50)
            {
                chart2.Series["AccX"].Points.RemoveAt(0);
                chart2.Series["AccY"].Points.RemoveAt(0);
                chart2.Series["AccZ"].Points.RemoveAt(0);
                borrar++;
            }
        }

        if (this.chart3.InvokeRequired)
        {
            SetChartCallback d = new SetChartCallback(ActualizarChart);
            this.Invoke(d, new object[] { });
        }
        else
        {
            chart3.Series["GyrX"].Points.AddXY(Time, GyrX);
            chart3.Series["GyrY"].Points.AddXY(Time, GyrY);
            chart3.Series["GyrZ"].Points.AddXY(Time, GyrZ);
            chart3.Titles[1].Text = "Modulo = " + Mod_Gyr.ToString() + " grados/seg";
            if (it >= 50)
            {
                chart3.Series["GyrX"].Points.RemoveAt(0);
                chart3.Series["GyrY"].Points.RemoveAt(0);
                chart3.Series["GyrZ"].Points.RemoveAt(0);
                borrar++;
            }
        }
    }

    string myPin = "1234";
    private bool pairDevice()
    {
        if (!deviceInfo.Authenticated)
        {
            if (!BluetoothSecurity.PairRequest(deviceInfo.DeviceAddress, myPin))
            {
                return false;
            }
        }
        return true;
    }

    private void chart2_Click_1(object sender, EventArgs e)
    {

    }

    private void chart3_Click(object sender, EventArgs e)
    {

    }

    bool ready = false;
```

```
        byte[] message;
        private void tbText_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == 13)
            {
                message = Encoding.ASCII.GetBytes(tbText.Text);
                ready = true;
                tbText.Clear();
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void chart1_Click(object sender, EventArgs e)
        {

        }

        private void chart2_Click(object sender, EventArgs e)
        {

        }
    }
}
```

9.6. Código en el Bioloid

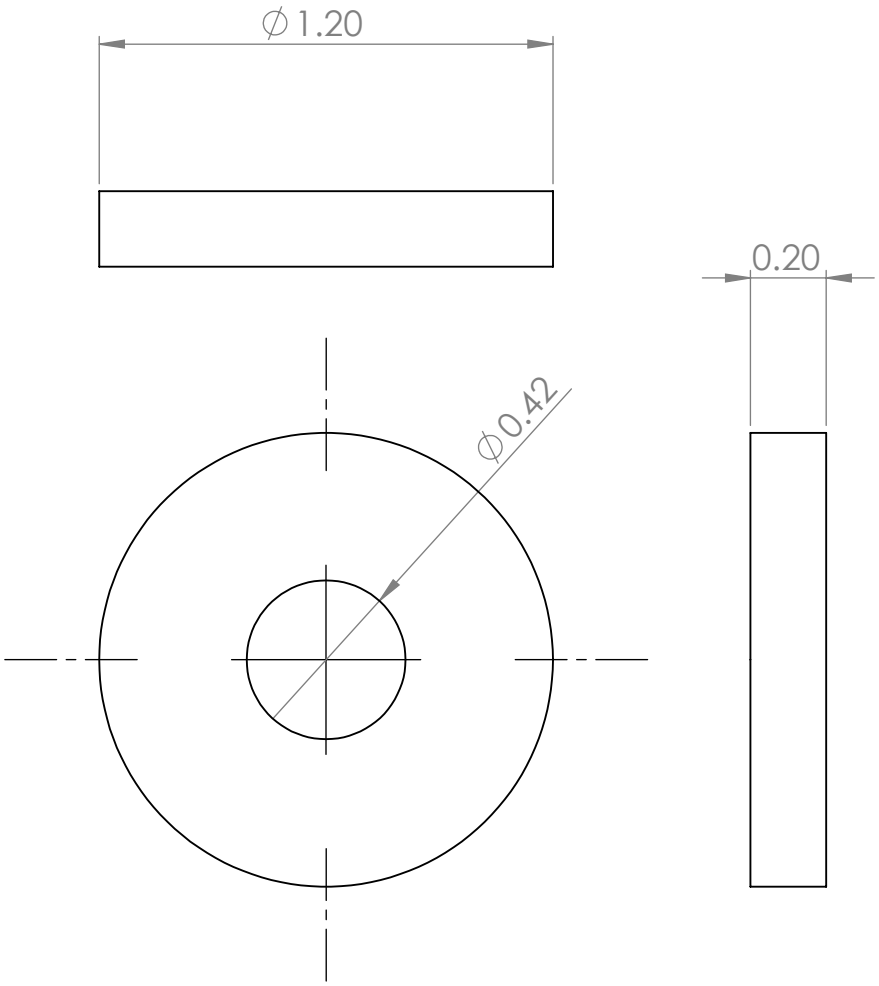
```
void USART1_Configuration(u32);
void Timer_Configuration(void);
void TimerInterrupt_1ms(void);
void RxD0Interrupt(void);
void DisableUSART1(void);
void ClearBuffer256(void);
byte CheckNewarrive_dxl(void);
void TxDByte_DXL(byte);
byte RxDByte_DXL(void);
void StartDiscount(s32);
byte CheckTimeOut(void);

/* Private functions ---------------------------------- Zigbee Communication -----------------------------------------------*/

void RxD2Interrupt(void);
void EnableZigbee(void);
void DisableZigbee(void);
byte CheckNewArrive_zgb(void);
void TxDByte_Zigbee(byte);
byte RxDByte_Zigbee(void);

/* Private functions ---------------------------------- sensor -------------------------------------------------*/
void ADC_Configuration(void);
uint16_t ReadAnalog(EPortA_t);

/* Private functions ---------------------------------- Common -------------------------------------------------*/
void RCC_Configuration(void);
void NVIC_Configuration(void);
void GPIO_Configuration(void);
void USART_Configuration(u8, u32);
```

```
void SysTick_Configuration(void);
void PrintCommStatus(int);
void PrintErrorCode(void);
void mDelay(u32);
void uDelay(u32);
void TxDString(byte*);
void TxDWord16(word);
void TxDByte16(byte);
void TxDByte_PC(byte);
void __ISR_DELAY(void);

/* Private functions ---------------------------------- Buzzer ----------------------------------------------*/
void start_countdown_buzzer(uint32_t);
void PlayNote(uint32_t, buzzed_note_t, uint8_t);
void Buzzed(uint32_t, uint32_t);
/* Private functions ----------------------------------  Leds  ----------------------------------------------*/
void SetLED(LED_t,uint8_t);

/* Motion Functions ------------------------------------------------------------*/
void Robot_ready();
void Robot_walk(int);
void Robot_subir_escalon();
void Robot_bajar_escalon();
void Robot_giro_derecha();
void Robot_giro_izquierda();
void Robot_giro_derecha_medio();
void Robot_giro_izquierda_medio();
void Robot_desplazamiento_derecha();
void Completed_movement_confirmation();
void Recepcion_confirmation();
void Robot_Giro(int);
void Robot_Posicion_angular();


/*******************************************************************************
 * Function Name  : main
 * Description    : Main program
 * Input          : None
 * Output         : None
 * Return         : None
 *******************************************************************************/
uint8_t ReadButton(Button_t button);

int j=1;

int main(void)
{
	/* ---------------------- Configuration: ------------------------------------ */
	/* System Clocks Configuration */
	RCC_Configuration();
	/* NVIC configuration */
	NVIC_Configuration();
	/* GPIO configuration */
	GPIO_Configuration();
	SysTick_Configuration();
	/* Timer configuration */
	Timer_Configuration();
	/* ADC configuration */
	ADC_Configuration();
	GPIO_ResetBits(PORT_SIG_MOT1P,PIN_SIG_MOT1P);
	GPIO_ResetBits(PORT_SIG_MOT1M,PIN_SIG_MOT1M);
	/* Puerto Zigbee inicializacion */
	dxl_initialize( 0, 1 );
```

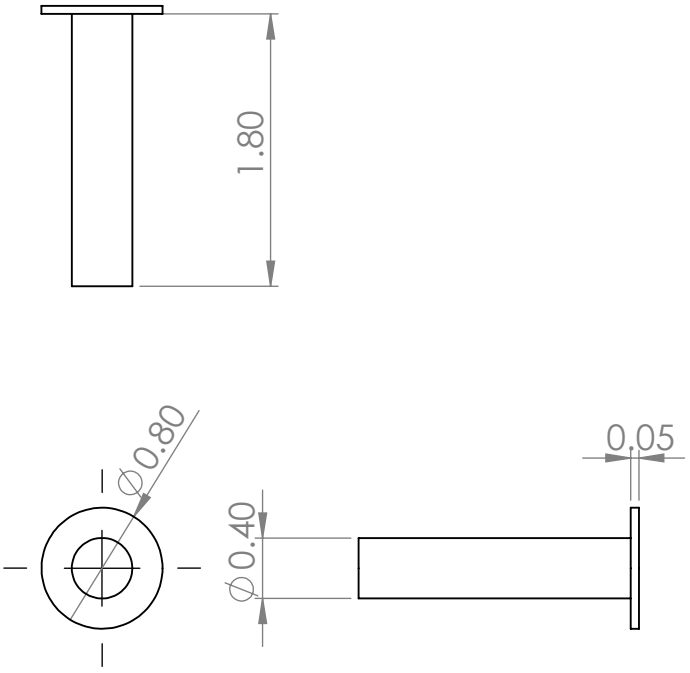
```
zgb_initialize(0); //Inicalización de la uart
EnableZigbee();
/* UART configuration */
USART1_Configuration(Baudrate_DXL);
USART_Configuration(USART_PC, Baudrate_PC);
/* Vector con los identificadores de cada motor */
for( i=0; i<NUM_ACTUATOR; i++ )
{
        id[i] = i+1;
}
/* Led de encendido */
GPIO_ResetBits(PORT_LED_POWER, PIN_LED_POWER);
/* Robot en posicion de espera */
Robot_ready();
mDelay(500);
/* ---------------------------------------------------------------------------- */

// Wait for START button to be pressed (and toggle MANAGE LED)
while (!ReadButton(START))
{
}

u8 LSB,MSB;
word RcvData, DATA=0;
GPIO_ResetBits(PORT_LED_AUX, PIN_LED_AUX);
mDelay(2000);
GPIO_SetBits(PORT_LED_AUX, PIN_LED_AUX);
while(DATA!=0xabdc){
        while(zgb_rx_check()!=1){
                zgb_tx_data(0x00b0);
        }
        RcvData= zgb_rx_data();
        LSB = dxl_get_lowbyte(RcvData);
        MSB = dxl_get_highbyte(RcvData);
        DATA  = (unsigned short)((MSB << 8) & 0xff00);
        DATA += LSB;
        mDelay(200);
}
zgb_tx_data(0x80a0);
zgb_tx_data(0x80a0);
zgb_tx_data(0x80a0);
GPIO_ResetBits(PORT_LED_AUX, PIN_LED_AUX);
mDelay(2000);
GPIO_SetBits(PORT_LED_AUX, PIN_LED_AUX);

while(1){
        while(zgb_rx_check()!=1){
                zgb_tx_data(0x00b0);
        }
                RcvData= zgb_rx_data();
                LSB = dxl_get_lowbyte(RcvData);
                MSB = dxl_get_highbyte(RcvData);
                DATA  = (unsigned short)((MSB << 8) & 0xff00);
                DATA += LSB;


        //GPIO_ResetBits(PORT_LED_AUX, PIN_LED_MANAGE);
        mDelay(500);
        switch(DATA){
                case 0x0502: //Giro derecha
                        Robot_giro_derecha(0,5);
                        GPIO_ResetBits(PORT_LED_PROGRAM, PIN_LED_PROGRAM);
                        mDelay(500);
```

```
                                        GPIO_SetBits(PORT_LED_PROGRAM, PIN_LED_PROGRAM);
                                        mDelay(500);

                                        DATA=0;

                                        break;
                                case 0x0904: //Giro Izquierda


                                        GPIO_ResetBits(PORT_LED_PLAY, PIN_LED_PLAY);
                                        mDelay(500);
                                        GPIO_SetBits(PORT_LED_PLAY, PIN_LED_PLAY);
                                        mDelay(500);
                                        Robot_giro_izquierda(0,5);
                                        DATA=0;

                                        break;
                                case 0x0308: //Adelante

                                        GPIO_ResetBits(PORT_LED_AUX, PIN_LED_AUX);
                                        //Robot_walk(5);
                                        mDelay(500);
                                        GPIO_SetBits(PORT_LED_AUX, PIN_LED_AUX);
                                        mDelay(500);
                                        Robot_walk(5);
                                        DATA=0;
                                        break;
                                default:
                                        //Robot_walk(5);
                                        GPIO_ResetBits(PORT_LED_MANAGE, PIN_LED_MANAGE);
                                        mDelay(500);
                                        GPIO_SetBits(PORT_LED_MANAGE, PIN_LED_MANAGE);
                                        mDelay(500);
                                        Robot_ready();
                                        break;
                        }
                        //Robot_ready();


                        //Robot_giro_derecha(0,5);
                        //Robot_giro_izquierda(0,5);
                        //Robot_Giro(90);
                        //Robot_walk(5);
                        //Robot_Posicion_angular(180);


                }

return 0;
}
```

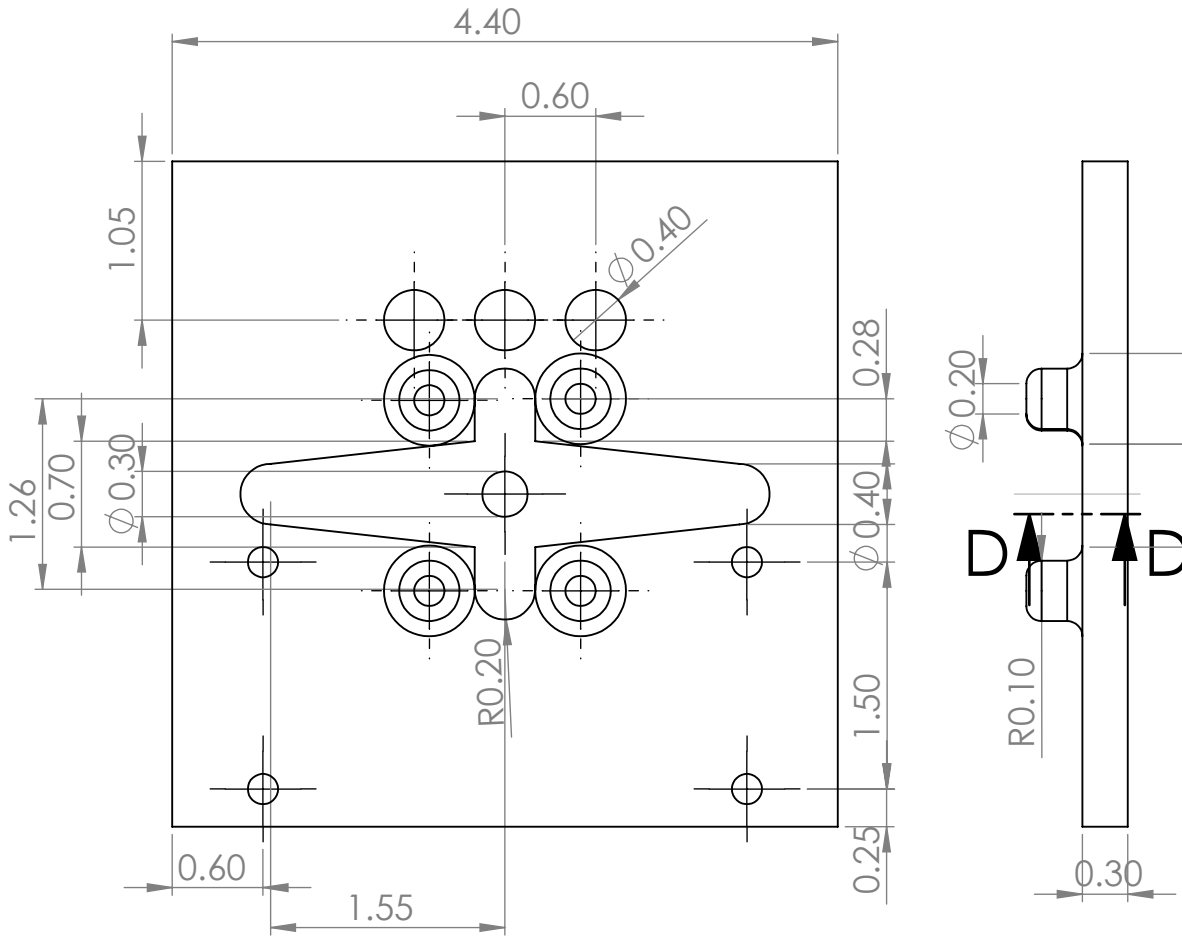
9.7. Planos del Soporte de la Cámara

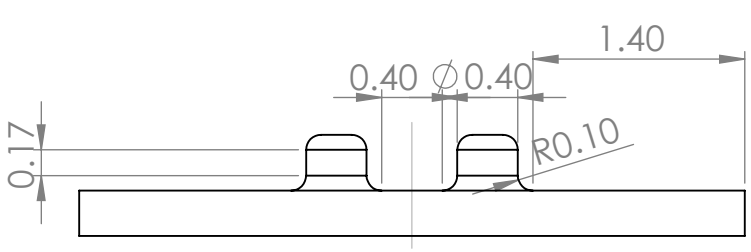


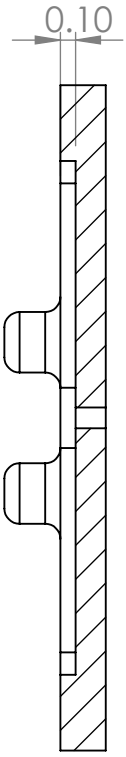
AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Arandela
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 1 — A4
		ESCALA:5:1 — HOJA 1 DE 1



AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Pasador
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 2 — A4
		ESCALA:2:1 — HOJA 1 DE 1

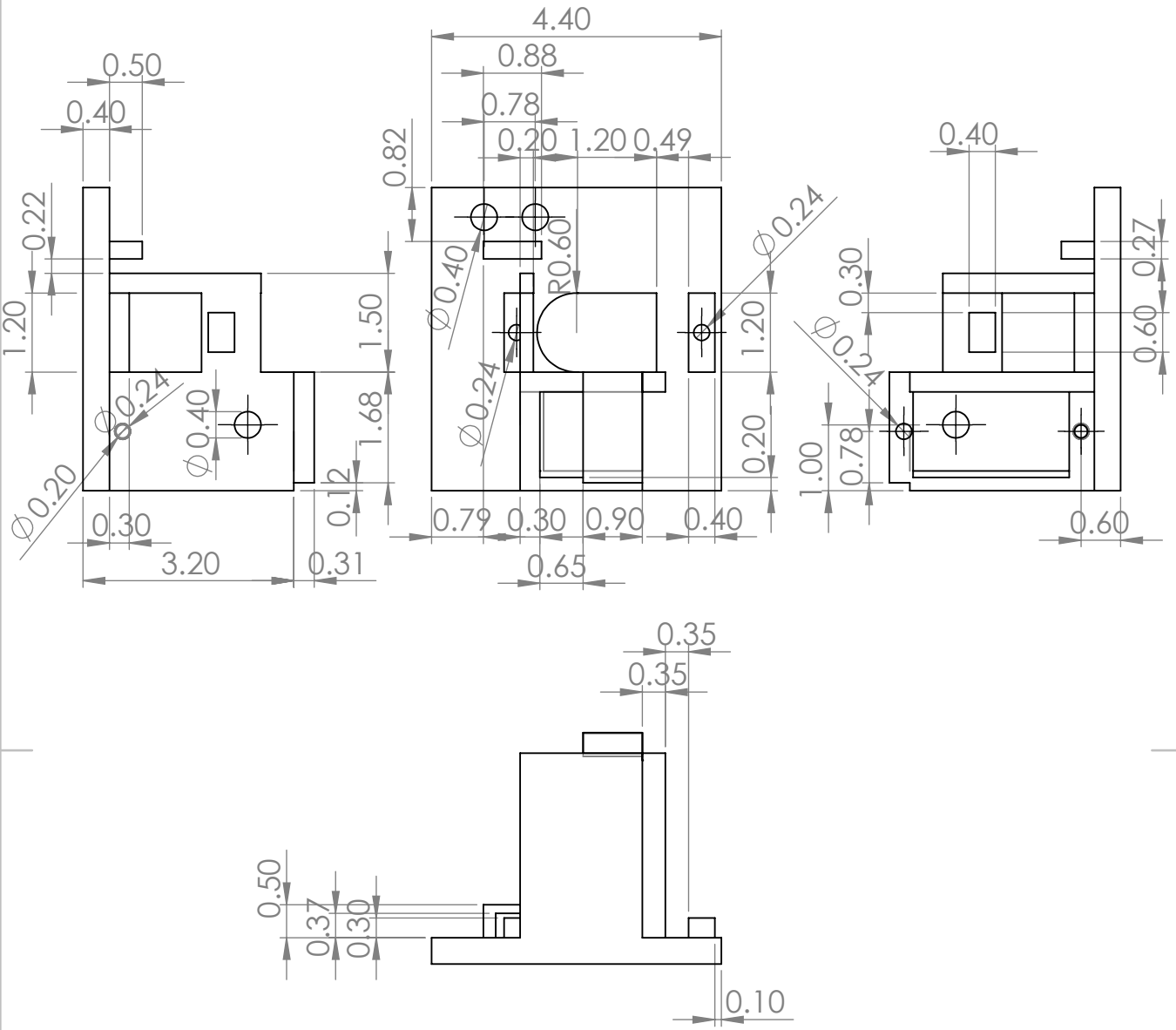




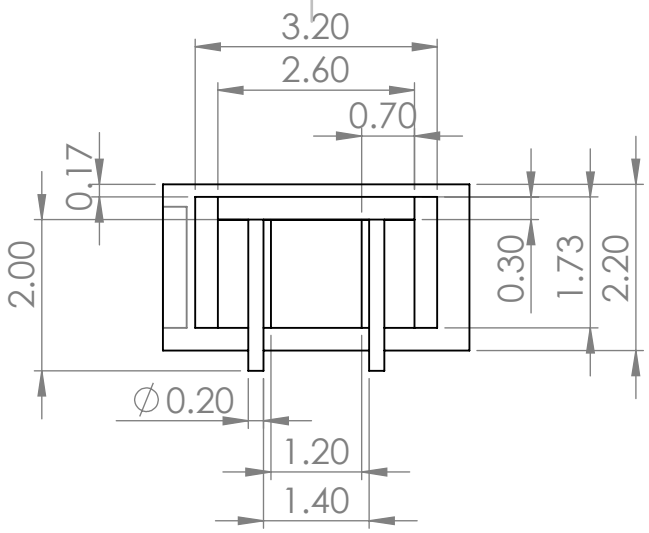


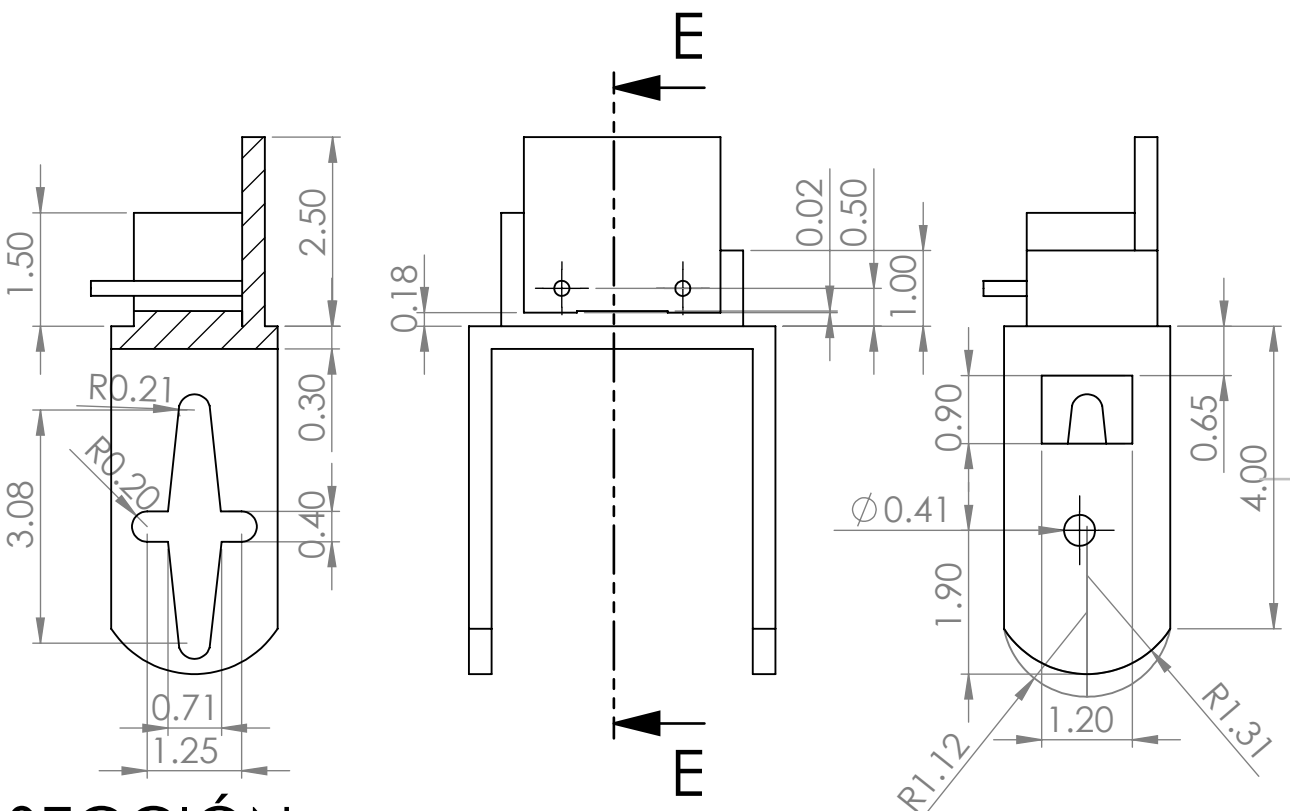
SECCIÓN D-D

AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Base del Soporte
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 3 — A4
		ESCALA:2:1 — HOJA 1 DE 1



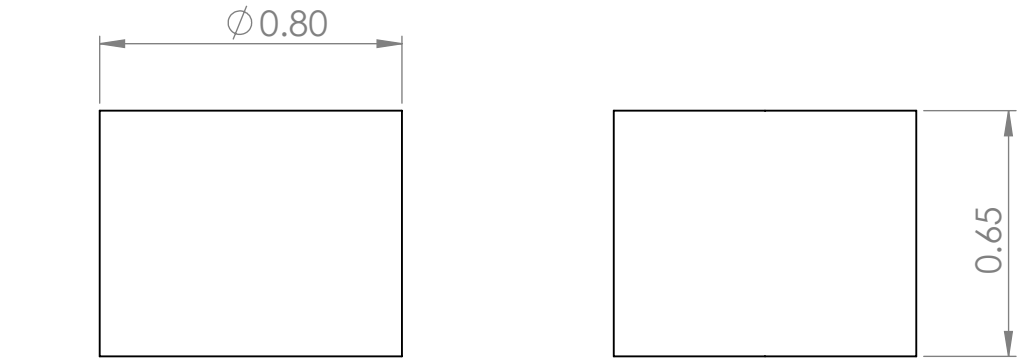
AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Pieza Central
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 4 — A4
		ESCALA:1:1 — HOJA 1 DE 1

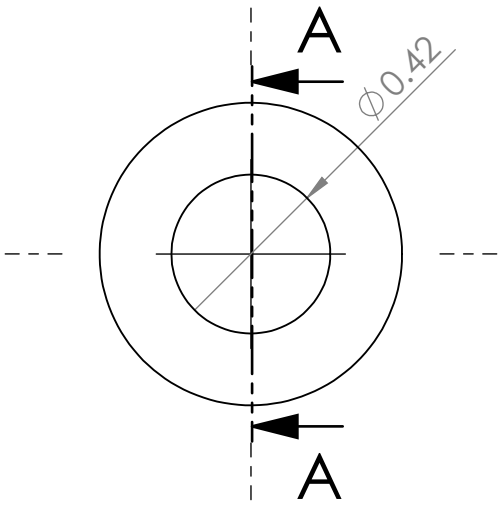


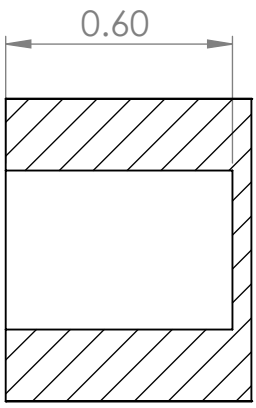


SECCIÓN E-E

AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Cabezal
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 5 — A4
		ESCALA:1:1 — HOJA 1 DE 1

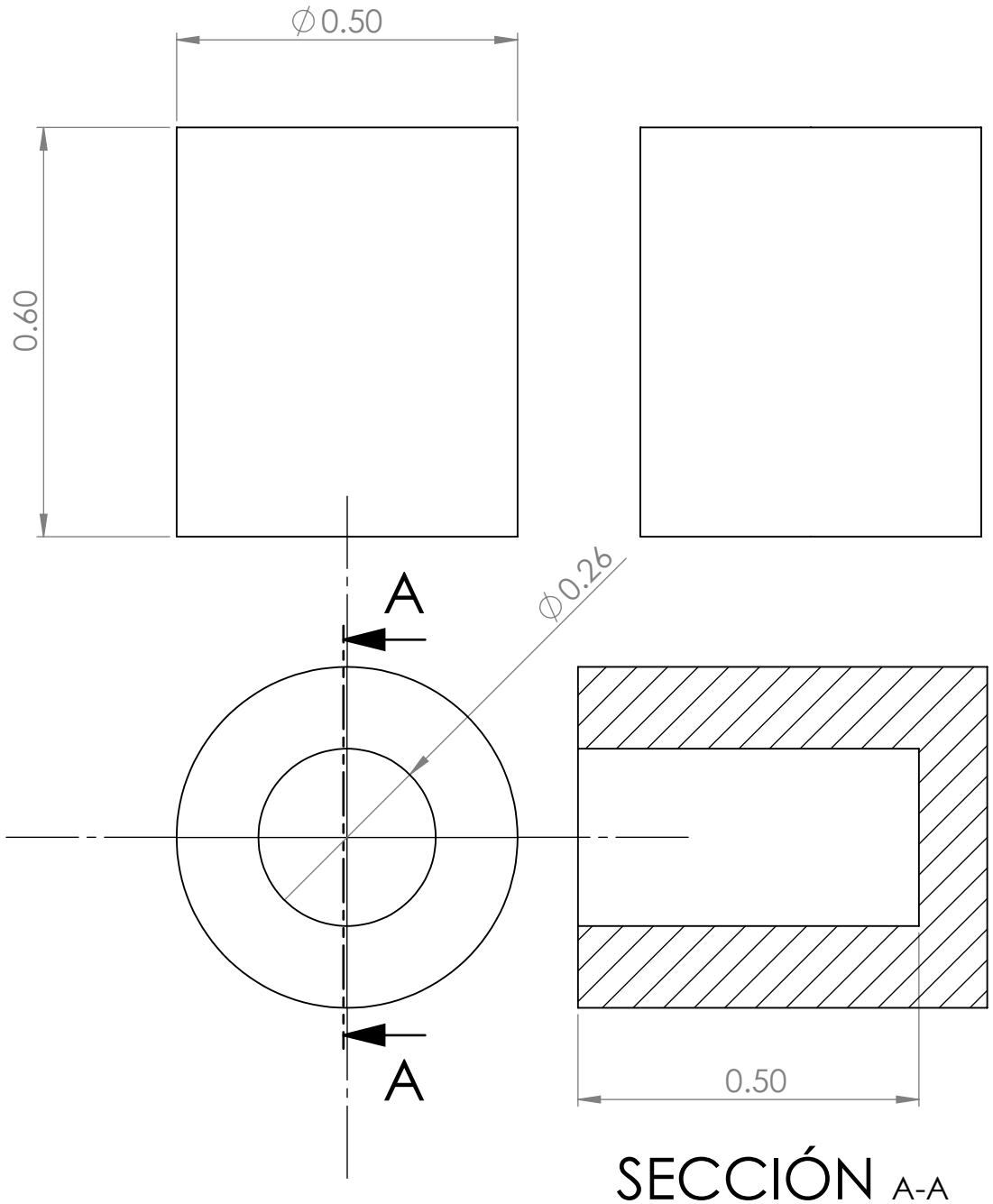






SECCIÓN A-A

AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Tapón 1	
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA		
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019		
	MATERIAL: PLA	N.º DE DIBUJO 6	A4
		ESCALA:5:1	HOJA 1 DE 1



SECCIÓN A-A

AUTOR: ANDRÉS MESEGUER VALENZUELA	UNIVERSIDAD POLITÉCNICA DE VALENCIA	TÍTULO: Tapón 2
PROYECTO: DESARROLLO DE UN SISTEMA DE VISIÓN ACTIVO EN ROBOT BIOLOID	NO CAMBIE LA ESCALA	
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN CM	FECHA: 16/09/2019	
	MATERIAL: PLA	N.º DE DIBUJO 7 — A4
		ESCALA:10:1 — HOJA 1 DE 1