*Publicaciones Técnicas* *Instituto de Astronomía*
*pubtecia@astro.unam.mx* *UNAM*

Reporte Técnico RT-2015-03

aceptación: 08-06-2015
recepción: 10-02-2015

# Diseño de un sistema de control para un telescopio de 11 pulgadas con montura Elevación-Azimut basado en una microcomputadora "BeagleBone Black" con sistema operativo Linux de tiempo real.

S. Zazueta, F. Murillo, G. Sierra, E. López, J.M. Núñez, J.M. Murillo, B. García.

Instituto de Astronomía. Universidad Nacional Autónoma de México.
Km. 103 Carretera Tijuana-Ensenada, Ensenada, B. C., México.

**RESUMEN:**

Este trabajo presenta el diseño e implementación de un sistema de control digital para motores, basado en un microcontrolador BeagleBone Black de tecnología reciente con un sistema operativo Linux de tiempo real. El sistema fue aplicado en el control de un telescopio de 11" con montura Elevación-Azimut. Con base en la pruebas de desempeño, concluimos que el diseño puede ser implementado en telescopios de mayor tamaño, como los existentes en San Pedro Mártir, adecuando la etapa de potencia.

## Contenido

## 1. INTRODUCCIÓN

Una de las líneas de trabajo del Departamento de Instrumentación Astronómica del Instituto de Astronomía es el desarrollo de sistemas de control de telescopios, tradicionalmente llamados "consolas". Desde la década de 1970, cuando se desarrolló la primera consola del telescopio de 2.1m del OAN, hasta la actualidad, se han desarrollado varias consolas y actualizaciones de las mismas. El avance en la tecnología ha permitido el desarrollo de consolas cada vez más compactas y robustas, con interfaces de operación amigables.

En el último año se logró el desarrollo de una consola compacta, con capacidad de control de telescopios de cualquier tamaño. Esto se puede lograr por medio de una implementación apropiada de la etapa de potencia. La consola está basada en una microcomputadora de tecnología reciente, modelo *BeagleBone Black* [1], que posee los recursos necesarios para implementar sistemas de control automático de servomecanismos.

La microcomputadora ejecuta el sistema operativo Linux en varias versiones: Debian, Angstrom, Ubuntu, así como la versión de tiempo real Debian-Xenomai, utilizada en sistemas de control de tiempo real, como es el caso de nuestra aplicación.

Tomando esta microcomputadora como base, se desarrolló la electrónica necesaria para el control de dos motores retroalimentados con codificadores incrementales. Se desarrollaron los programas de posicionamiento con algoritmos de control digital y también se desarrolló una interfaz de usuario.

El diseño completo fue evaluado inicialmente en un banco con un arreglo de un motor y codificador. El resultado fue satisfactorio por lo que posteriormente se instaló en un telescopio *Celestron* de 11 pulgadas, con montura Elevación-Azimut (ver *Figura 1*), del cual utilizamos sus motores, codificadores y mecánica original.


***Figura 1:*** *Telescopio Celestron en el Laboratorio de Electrónica, IAUNAM-Ensenada.*

El presente documento describe el diseño de la nueva consola y su implementación en el telescopio de 11 pulgadas. Las pruebas de movimiento indicaron que es posible implementar el sistema de control en los telescopios del Observatorio Astronómico Nacional en San Pedro Mártir.

## 2. DESCRIPCIÓN DEL SISTEMA

El sistema de control de posición del telescopio consta de una microcomputadora *BeagleBone* y dos tarjetas electrónicas diseñadas para el manejo de dos motores retroalimentados con codificadores incrementales. Las tarjetas fueron diseñadas y construidas en el Laboratorio de Electrónica del Instituto de Astronomía en Ensenada.

La *Figura 2* muestra un diagrama a bloques del sistema desde el punto de vista electrónico. Como pieza fundamental tenemos la **Microcomputadora BeagleBone** que se conecta a la **Tarjeta de Interfaz**. Ésta se encarga de distribuir y acondicionar las señales de los puertos de la microcomputadora hacia los demás componentes del sistema. En el Apéndice A se muestra el diagrama esquemático y el mapa de componentes de esta tarjeta.

La **Tarjeta de Potencia de Motores** consta de dos amplificadores de corriente basados en puente H, especialmente construidos para el movimiento de motores de corriente continua con escobillas. Los amplificadores son controlados por medio de modulación de ancho de pulso comúnmente denominada PWM (por sus siglas en inglés). La Tarjeta de Potencia también cuenta con cuatro entradas opto-acopladas para leer interruptores de referencia o límites, de las cuales sólo se utilizan dos entradas como señales de referencia o inicio: se usa una entrada para el interruptor de inicio del Eje de Elevación y otra para el de Azimut. A esta tarjeta se conectan directamente los motores. Su diagrama esquemático y mapa de componentes se muestran en el Apéndice B.

Las señales de los codificadores son reforzadas y acondicionadas en la Tarjeta de Interfaz, e introducidas a la Microcomputadora.

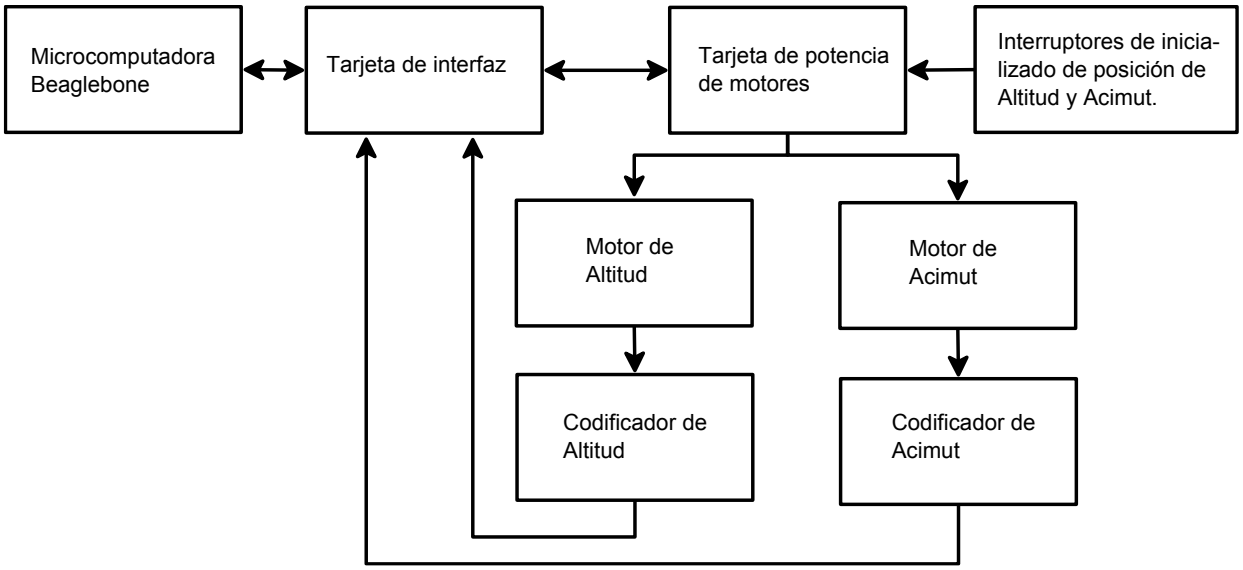



***Figura 2:*** *Diagrama a bloques del Sistema de Control de Posición.*

La *Figura 3* muestra una fotografía del sistema de control electrónico montado en uno de los brazos del telescopio *Celestron* de 11". Como se observa, su tamaño compacto permite que se instale en ese espacio; una vez colocada la tapa original del brazo, la electrónica queda oculta y protegida.

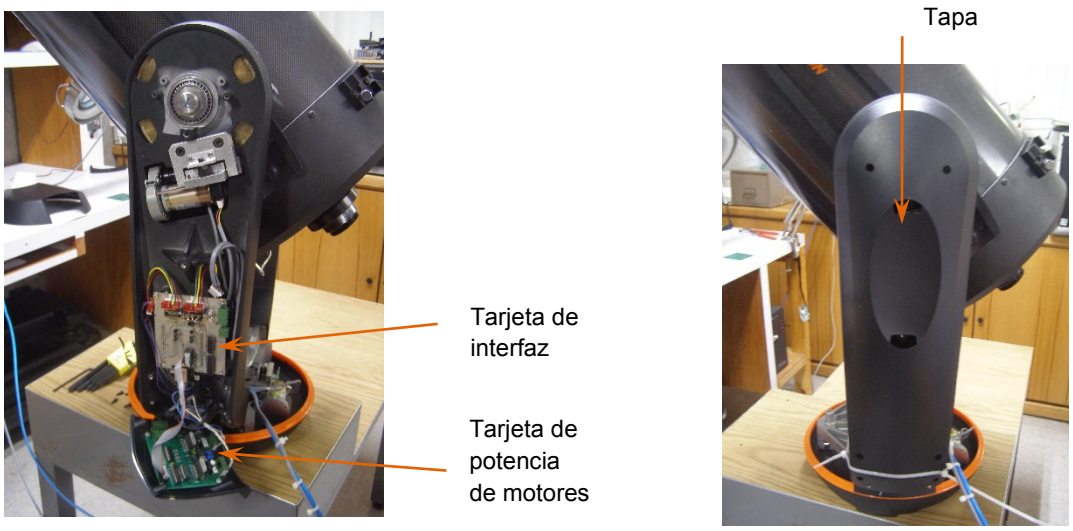



***Figura 3:*** *Electrónica de control instalada en un brazo de la montura del telescopio.*

### 2.1. PARÁMETROS DEL TELESCOPIO

La Tabla 1 muestra los parámetros del telescopio medidos en el laboratorio.

**TABLA 1**

Parámetros del telescopio.

| | |
|---|---|
| Pulsos por segundo de arco en ambos ejes | 8.95061728395 |
| Velocidad máxima en ambos ejes | 2.5 grados por segundo |
| Pulsos en 360 grados en ambos ejes | 11,600,000 |

## 3. DISEÑO DE PIEZAS MECÁNICAS

Para limitar la carrera de movimiento e inicializar la posición de los ejes de Elevación y Azimut, se diseñaron mecanismos y monturas nuevas.

En el eje de movimiento de Azimut se utilizaron tres micro-interruptores colocados sobre la base del telescopio, dos para limitar la carrera de movimiento y uno para inicializar. A estos interruptores se les diseñó una montura con 3 grados de libertad para su ajuste. Para activar los interruptores se diseñó un mecanismo ajustable de tres bielas montadas sobre el eje de giro. Dos de las bielas activan los micro-interruptores que limitan el movimiento de giro, y una tercera biela, en forma semicircular, activa el micro-interruptor de inicializado (ver *Figura 4*).

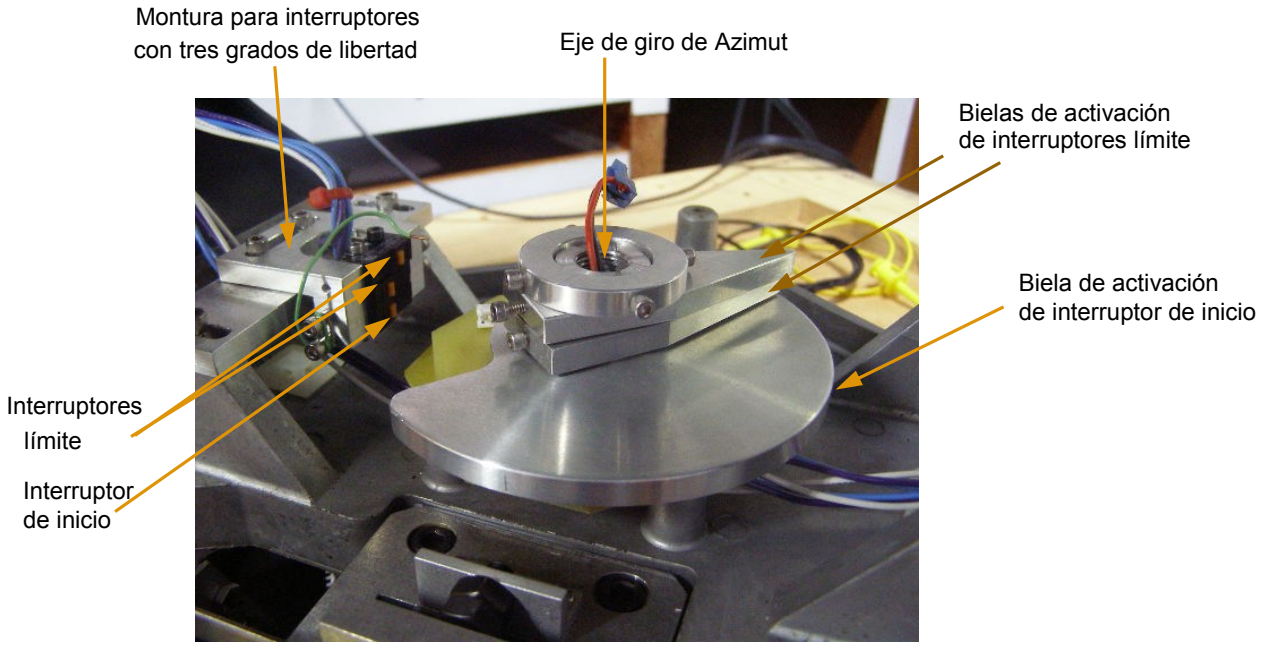



***Figura 4:*** *Piezas mecánicas diseñadas para el eje de azimut.*

En el eje de movimiento de elevación, al igual que en el de azimut, se diseñaron mecanismos y monturas. Se usaron tres micro-interruptores de corriente. Dos para limitar la carrera de movimiento y el tercero para inicializar. Se diseñó una montura para los interruptores límite y se colocó una leva que los activa. Se diseñó una media luna que activa el interruptor de inicio (ver *Figura 5*).

En el Apéndice C se muestran los planos de todas las piezas que fueron diseñadas y construidas.

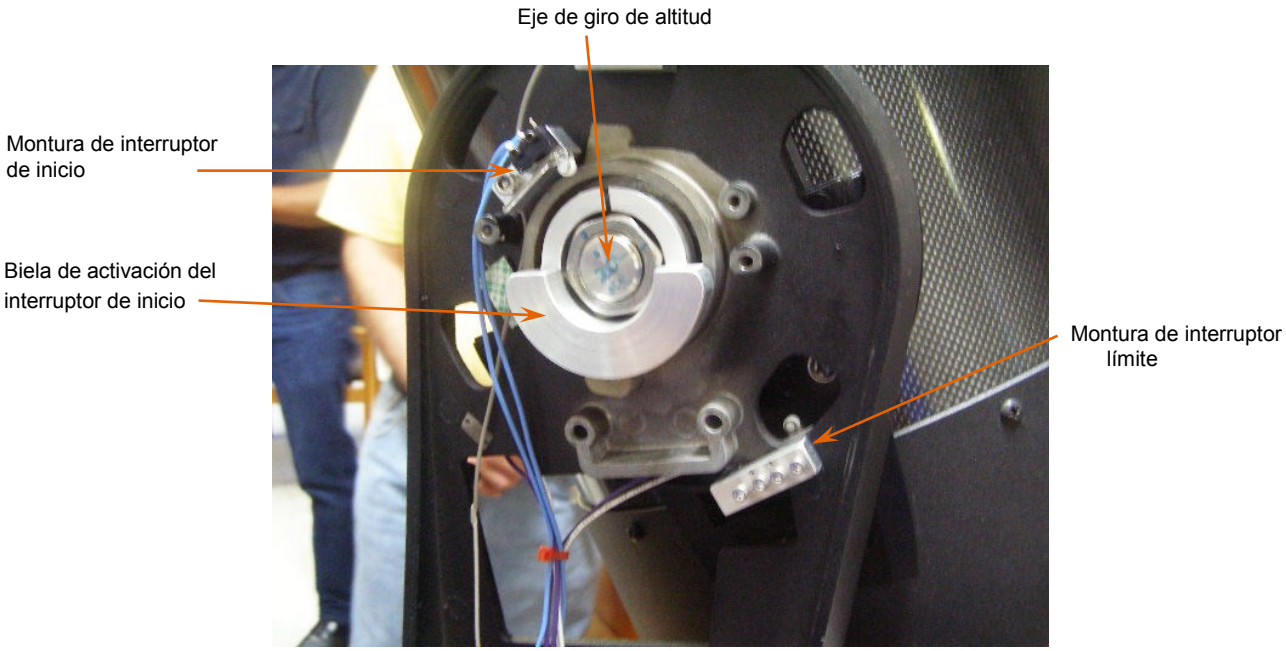



***Figura 5:*** *Piezas mecánicas diseñadas para el eje de elevación.*

## 4. PROGRAMAS DESARROLLADOS

Para el control de los motores del telescopio se desarrollaron programas de bajo y alto nivel. La *Figura 6* muestra un diagrama con la distribución de los programas. En el microcontrolador *BeagleBone* se ejecutan dos programas: uno de bajo nivel denominado **servobb** y otro de alto nivel denominado **consteld**. El programa **servobb** hace uso de la biblioteca de manejo de recursos de hardware denominada **servo_lib** desarrollada para esta aplicación.

Para la operación amigable, se desarrolló un programa de interfaz de usuario denominado **consola**, que se ejecuta en la computadora de usuario. Este programa puede recibir instrucciones del programa de código abierto *Stellarium*.

En las siguientes secciones se describe cada uno de los programas.

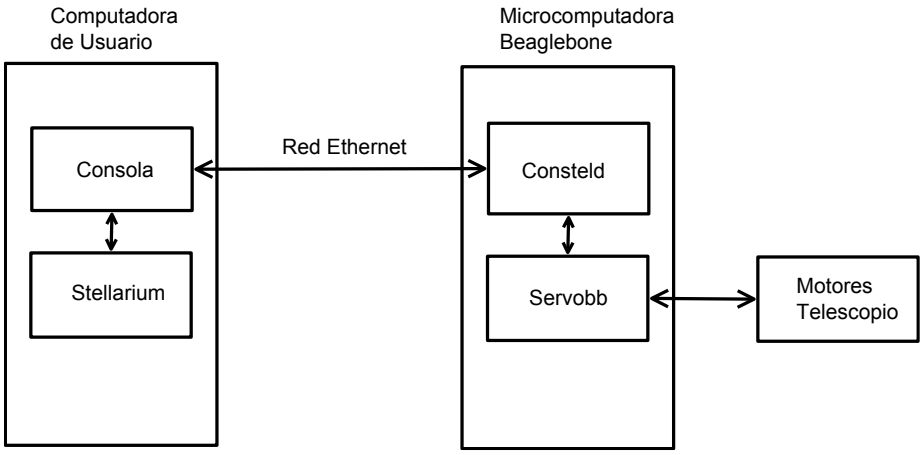



***Figura 6:*** *Distribución de los programas desarrollados.*

#### 4.1. EL MANEJADOR DE BAJO NIVEL

El telescopio cuenta con dos ejes de movimiento: Elevación y Azimut, cada uno equipado con un motor de corriente continua y un codificador de posición de tipo incremental con salida en cuadratura. Para el control de posición de cada eje, es necesario tener acceso desde la computadora a una acción de control y a la señal de retroalimentación, en este caso la posición del motor.

Dentro de los recursos de hardware disponibles en el *BeagleBone,* se encuentran los subsistemas de modulación de ancho de pulso PWMSS. Con estos es posible controlar un amplificador de corriente que sería la acción de control. El monitoreo de la posición lo obtenemos mediante los módulos lectores del codificador en cuadratura EQEP, que forman parte de los subsistemas PWMSS del *BeagleBone*.

El telescopio cuenta también con dos interruptores de referencia para inicializar cada eje. Estos interruptores son monitoreados a través de dos entradas de propósito general GPIO de la microcomputadora.

Los subsistemas PWMSS, GPIO y los módulos EQEP son configurables a través de una serie de registros. Esto se hace desde funciones desarrolladas y agrupadas en una biblioteca denominada **servo_lib.c**, escrita en lenguaje C. Esta biblioteca constituye el manejador o *driver* de bajo nivel para la tarjeta de interfaz.

Todos los recursos de hardware del *BeagleBone* utilizados en esta aplicación son manejados desde sus funciones. La Tabla 2 muestra las funciones disponibles en la biblioteca y su funcionalidad.

**TABLA 2**

Funciones en la biblioteca **servo_lib.c**.

| | |
|---|---|
| configura() | • Realiza un mapeo de memoria de los registros correspondientes a los subsistemas PWMSS1, PWMSS2 y puertos de entrada-salida GPIO0 y GPIO1.<br>• Habilita los módulos PWMSS1 y PWMSS2 configurando los registros CM_PER_EPWMSS1_CLKCTRL y CM_PER_EPWMSS2_CLKCTRL.<br>• Habilita los puertos configurando los registros CM_WKUP_GPIO0_CLKCTRL y CM_WKUP_GPIO1_CLKCTRL.<br>• Habilita los módulos EQEP1 y EQEP2 escribiendo en los registros QEPCTL y QDECCTL. |
| pwm_frec(int frec) | Define la frecuencia de las señales de salida PWM1 y PWM2. Para hacerlo modifica el registro TBPRD. El parámetro "frec" está definido en Hertz. |
| pwm_set_motor1(int p) | Define el porcentaje de estado activo de la señal PWM1. Para hacerlo escribe en el registro CMPA. El parámetro "p" se introduce en valores de 0 a 100. Con esta función se controla la velocidad del motor. |
| pwm_set_motor2(int p) | Define el porcentaje de estado activo de la señal PWM2. Para hacerlo escribe en el registro CMPB. El parámetro "p" se introduce en valores de 0 a 100. |
| gira_motor1(int s) | Habilita el giro del motor 1 en sentido s. |
| gira_motor2(int s) | Habilita el giro del motor 2 en sentido s. |
| alto_motor1() | Detiene el movimiento del motor 1, independiente al estado de la señal PWM1. |
| alto_motor2() | Detiene el movimiento del motor 2, independiente al estado de la señal PWM2. |
| lee_sw0_1() | Regresa 0 o 1 correspondiente al estado del puerto de entrada GPIO014. |
| lee_sw1_1() | Regresa 0 o 1 correspondiente al estado del puerto de entrada GPIO030. |
| lee_sw2_1() | Regresa 0 o 1 correspondiente al estado del puerto de entrada GPIO031. |
| lee_sw3_1() | Regresa 0 o 1 correspondiente al estado del puerto de entrada GPIO05. |
| limpia_banderas_codificador_1() | Limpia el registro de banderas QCLR del codificador 1. |
| limpia_banderas_codificador_2() | Limpia el registro de banderas QCLR del codificador 2. |
| lee_codificador_1() | Regresa el valor del contador QPOSCNT del codificador 1. |
| lee_codificador_2() | Regresa el valor del contador QPOSCNT del codificador 2. |
| inicializa_codificador_1(int posicion) | Escribe en el registro contador QPOSCNT del codificador 1, el valor dado en el argumento "posición". |
| inicializa_codificador_2(int posicion) | Escribe en el registro contador QPOSCNT del codificador 2, el valor dado en el argumento "posición". |
| habilita_oe() | Habilita el registro de señales de salida. |

Esta biblioteca se compila como un objeto que puede ser incluido en el programa de aplicación y que puede estar desarrollado en otro lenguaje de programación, como Python.

Dado que la tarjeta de interfaz es de propósito general, diseñada para el control de movimiento de dos motores de corriente continua, las funciones tienen nombres genéricos donde se incluyen las palabras "motor_1", "motor_2", "codificador_1", "codificador_2". La Tabla 3 muestra la relación entre esta nomenclatura y la de nuestra aplicación particular, que es el control de movimiento de un telescopio.

**TABLA 3**

Asignación de motores, codificadores e interruptores.

| | |
|---|---|
| Motor 1 | Motor del eje de elevación |
| Motor 2 | Motor del eje de azimut |
| Codificador 1 | Codificador del eje de elevación |
| Codificador 2 | Codificador del eje de azimut |
| sw0_1 | Interruptor de origen del eje de elevación |
| sw1_1 | Interruptor de origen del eje de azimut |

### 4.2. PROGRAMAS DE CONTROL DE TELESCOPIO

Se usan dos programas para realizar el control del telescopio:

1. Un programa servidor con tareas de tiempo real. Este programa realiza los algoritmos de control digital de los motores. A este programa lo llamamos **servobb**.
2. El segundo programa interpreta las instrucciones de alto nivel y genera instrucciones de bajo nivel para el programa de control de motores. A este programa lo llamamos **consteld.py**.

Las instrucciones de alto nivel se reciben a través de un puerto de *socket* de tipo TCP/IP.

#### 4.2.1. PROGRAMA SERVOBB

Para implementar un controlador digital de servomotores se requiere un sistema operativo de tiempo real. Escogimos de manera natural, como base, el Linux con extensiones de tiempo real Xenomai.

La elección se basó en la disponibilidad en la red internet de distribuciones dedicadas basadas en Debian para el *BeagleBone Black* (BBB) que ya cuentan con las extensiones de tiempo real necesarias. Esto ahorra una considerable cantidad de tiempo de desarrollo al no tener que reconfigurar y recompilar el núcleo de Linux.

El programa de tiempo real ejecuta dos hebras en paralelo; una de las hebras realiza los algoritmos de control digital.

Básicamente la tarea de tiempo real ejecuta las siguientes operaciones:

- Lectura de los valores de posición de los codificadores.
- Lectura de los sensores de posición de inicio.
- Cálculo de trayectorias y transformación de coordenadas.
- Cálculo de algoritmos de control digital (en este caso usamos un algoritmo PID con optimizaciones).
- Verificación de rutinas de seguridad.
- Actualización de las acciones de control.

La hebra de tiempo real se ejecuta cada milisegundo. Es decir, la frecuencia de muestreo del controlador digital es de 1 KHz. Los cálculos de transformación de coordenadas se realizan a una frecuencia de 100 Hz.

En paralelo con la hebra de tiempo real, se ejecuta la hebra de comunicación. Ésta se encarga de recibir instrucciones por medio de un *socket* de tipo archivo UNIX.

En general este programa recibe instrucciones de bajo nivel y tienen que ver con el manejo de las variables de control y monitoreo del funcionamiento de los servomotores.

Se utilizó la biblioteca **comoan** [2] para implementar la parte del servidor y de ejecución de instrucciones.

Los algoritmos de control y toda la parte de tiempo real se basan en los programas desarrollados para el control de los telescopios del OAN [3, 4, 5].

Las tareas de tiempo real son *tasklets* que usan funciones de la biblioteca Xenomai.

Se usan las funciones definidas en la Sección 4.1 para el acceso a los valores a los contadores de cuadratura, valores de entrada y el control de los amplificadores PWM.

Las instrucciones del programa **servobb** se listan en la Tabla 4. Debido a que la tarjeta de interfaz sirve para controlar 2 motores, la terminación “X” o “Y” se usa para diferenciar a qué motor corresponde la instrucción. Así que sólo se presentan las instrucciones para el motor X y las que afectan la operación de ambos motores.

**TABLA 4**

Instrucciones del programa **servobb**

| Instrucción | Función |
|---|---|
| KPX n.n | Cambia el parámetro KP (constante proporcional) del controlador digital PID del motor "X". El valor del parámetro es un número real "n.n".<br>KPY lo hace para el motor "Y". |
| KDX n.n | Cambia el parámetro KD (constante derivativa) del controlador digital PID del motor "X". |
| KIX n.n | Cambia el parámetro KI (constante integral) del controlador digital PID del motor "X". |
| ILX n.n | Cambia el parámetro IL (límite de integración) del controlador digital PID del motor "X". |
| UMAXX n.n | Cambia el parámetro UMAX (valor máximo de la acción de control) del controlador del motor "X". |
| CONTROL_PX | Selecciona como control digital a un controlador tipo P para el motor X. Sólo utiliza la parte proporcional. La acción de control "u" está definida por:<br>u = KP * e<br>e = referencia – posición actual |
| CONTROL_PIX | Usa el controlador PI para el motor X. La acción de control está definida por:<br>u= KP*e + min( KI * Sumatoria(e) , IL ) |
| CONTROL_PIDX | Usa el controlador PID para el motor X. La acción de control está definida por:<br>u= KP*e + min( KI * Sumatoria(e) , IL ) + KD*deltaE<br>donde:<br>deltaE = error actual – error anterior. |
| DAX n | Ajusta la salida del PWM del motor X al valor "n". Abre el lazo de control. |
| RST | Restablece todas las variables del controaldor a cero. Abre el lazo de control. |
| RST_S | Restablece el estado del controlador X. Ajusta a cero el valor de la referencia. No abre el lazo de control. |
| X= n | Cambia el valor de la referencia del motor X al valor "n". Donde "n" es un parámetro de tipo entero. |
| XR= n | Suma a la referencia actual del motor X el valor "n". Donde "n" es un parámetro de tipo entero. Es lo que se conoce como un cambio de relativo. |

| Instrucción | Función |
|---|---|
| MAXPOSX n | Cambia el valor máximo permitido de la posición del motor X. Si el motor se mueve más allá de esta posición se abre el lazo de control. |
| MINPOSX | Cambia el valor mínimo permitido de la posición del motor X. Si el motor se mueve más allá de esta posición se abre el lazo de control. |
| ERROR_MAXX n | Cambia el valor absoluto del error máximo tolerado antes de abrir el lazo de control. Este parámetro es útil cuando se genera una pérdida de la señal de retroalimentación o cuando se atora el motor por alguna falla mecánica. |
| VX= n.n | Cambia el valor de la velocidad deseada del motor X. El parámetro tiene como unidades (pulsos del codificador)/muestreo. El muestreo es de 1 KHz. |
| AX= n.n | Cambia el valor de la aceleración deseada del motor X. El parámetro tiene como unidades (pulsos del codificador)/(muestreo/muestreo). |
| VELSEGX= n.n | Cambia el valor de la velocidad de seguimiento deseada del motor X. El parámetro tiene como unidades (pulsos del codificador)/muestreo. |
| HABSEGX | Habilita el seguimiento del motor X. Lo que se conoce como modo velocidad o "jog". El motor se empieza a mover de forma constante a la velocidad definida por el parámetro "VELSEGX=". |
| DESHABSEGX | Deshabilita el modo velocidad del motor X. |
| CGANX n1.n1 n2.n2 | Cambia el valor del multiplicador de la acción de control. El propósito de estos parámetros es definir una zona donde la acción de control se puede modificar para lograr mejor control. La amplitud de la zona está definida por el parámetro "n1.n1" y el cambio de ganancia o multiplicador está definido por el parámetro "n2.n2".<br>La acción de control se define por:<br>$un = u*(1.0 + f(n1.n1,n2.n2,e))$<br>donde :<br>un = acción de control modificada.<br>u= acción de control que sale del controlador digital.<br>e= error actual.<br>donde la función "f" está definida por:<br>$g = 10.0 *(1 - e/n1.n1)$<br>$f = n2.n2 * (a*g*g + b*g + c)/ 10.0$<br>Los parámetros a,b,c definen una función cuadrática suave. |
| BITIX n | Máscara del bit de entrada que se asocia al inicio o "HOME" del motor X. El valor "n" debe ser una potencia de 2. |

| Instrucción | Función |
|---|---|
| BBIX n | Si el parámetro es diferente de cero habilita la búsqueda de un cambio en el estado de los "bits" de entrada del motor X. La bandera de inicio afectada está definida por el parámetro BITIX. Por si sola la instrucción BBIX no realiza ningún movimiento. Así pues para encontrar el "inicio" hay que mover el motor a alguna posición y monitorear el estado de las banderas del motor para saber cuándo hubo un cambio.<br>Cuando se detecta un cambio del estado de las entradas de inicio se ajusta el valor de la variable PIX al valor donde se detectó el cambio de estado.<br>También cambia al valor de la variable BIX a 0x800. Ver la instrucción ESTADO. |
| ESTADO | Reporta el estado de algunas variables de los motores X e Y.<br>El formato de reporte es:<br>X=n.n XD=n.n XT=n.n CTX=n UX=n BIX=oxh PIX=n.n<br>Y=n.n YD=n.n YT=n.n CTY=n UY=n BIY=oxh PIY=n.n<br>Donde:<br>X= posición actual del motor X.<br>XD= referencia del motor X.<br>XT= posición actual generada por el generador de trayectorias del motor X.<br>CTX= tipo de control:<br>• 0=lazo abierto<br>• 1=control proporcional o control P.<br>• 2= control PI.<br>• 4= control PID.<br>UX= acción de control del motor X.<br>BIX= estado de la máquina de estados de búsqueda de inicio. Está reportado en hexadecimal, el valor empieza con "ox" para denotarlo. Los estados de BIX son:<br>• 0x1 = Indica el inicio del proceso de búsqueda de inicio.<br>• 0x2 = Indica que ajustó sentido de búsqueda de inicio.<br>• 0x80 = Buscando inicio en sentido positivo.<br>• 0x100 = Buscando inicio en sentido negativo.<br>• 0x800 = Encontró el inicio.<br>PIX= Posición donde se encontró el inicio.<br>Los parámetros del motor Y son similares. |

#### 4.2.1.1. INSTRUCCIONES PARA EL MANEJO DE COORDENADAS (SERVOBB)

Se pueden manejar dos tipos de coordenadas que se asocian a los ejes A y B del programa **servobb**. El programa ejecuta los cálculos de coordenadas en tiempo real. A esta parte del programa la llamamos hebra de coordenadas o "HDC".

El primer tipo de coordenadas es lineal y se asocia directamente al eje A con el motor X, y al eje B con el motor Y. Con este tipo de coordenadas se puede manejar un telescopio con montura ecuatorial. Inclusive es posible manejar una montura ecuatorial con las instrucciones definidas en el apartado anterior.

El segundo tipo de coordenadas implica una transformación para el manejo de coordenadas Azimut-Elevación. El programa **servobb** ejecuta la transformación de:

(ÁNGULO HORARIO, DECLINACIÓN) → (AZIMUT, ELEVACIÓN)

El eje de AZIMUT → eje X
El eje de ELEVACIÓN → eje Y

Las instrucciones para el manejo de coordenadas se definen a continuación.

**TABLA 6**

Instrucciones para el manejo de coordenadas

| Instrucción | Función |
|---|---|
| HAB_A | Habilita la ejecución de la hebra de coordenadas (HDC) para el eje A. |
| DESHAB_A | Deshabilita la ejecución de la HDC para el eje A. |
| HAB_B | Habilita la ejecución de la HDC para el eje B. |
| DESHAB_B | Deshabilita la ejecución de la HDC para el eje B. |
| AH n.n | Ajusta el valor del ángulo horario deseado al parámetro n.n. El parámetro está definido en radianes. |
| DEC n.n | Ajusta el valor de declinación deseado al parámetro n.n. El parámetro está definido en radianes. |
| LATITUD n1.n1 | Define el parámetro de la latitud del lugar. Está en radianes. |
| EJEA n.n n1.n1 | Ajusta los parámetros de las constantes de proporción y la velocidad de seguimiento el eje. El parámetro n.n define el número de pulsos del codificador por radian.<br>El parámetro n1.n1 define la velocidad de seguimiento del eje en radianes por muestreo/10. Cuando se usa la instrucción "VELSEGA=" se debe poner el parámetro n1.n1 en cero. |
| EJEB n.n n1.n1 | Ver EJEA. |
| VELSEGA= n.n | Ajusta el valor de la velocidad de seguimiento del eje A. El parámetro está en radianes por muestreo/10 es decir cada 10 milisegundos. |
| HABSEGA | Habilita el seguimiento del eje A. |
| DESHABSEGA | Deshabilita el seguimiento del eje A. |
| HABSEGB | Habilita el seguimiento del eje B. |

| | |
|---|---|
| DESHABSEGB | Deshabilita el seguimiento del eje B. |
| COORDSA n1.n1 n2.n2 | Especifica las coordenadas de ángulo horario y declinación y habilita la HDC en modo de coordenadas azimut-elevación. Habilita el seguimiento o guiado. |
| NOGUIAA | Quita el guiado. |
| SPOFFA n1.n1 n2.n2 | Estos parámetros especifican un desplazamiento que se agrega a las posiciones deseadas de los ejes A y B. Se dan en radianes. La posición (pdA,pdB) a la que desplazarán los ejes está definida por<br>pdA = ah - n1.n1<br>pdB = dec – n2.n2<br>Esto es útil para definir una posición de reposo ya que los movimientos de los ejes son relativos a esta posición. |
| MAXMINACIMELEV n1 n2 n3 n4 | Define los valores máximos y mínimos a los que se puede mover el telescopio. Los valores están en radianes. Es útil para no llegar a las posiciones límites, en particular para no permitir que se toque un interruptor límite.<br>n1 = valor máximo del eje de azimut<br>n2= valor mímino de azimut.<br>n3= valor máximo del eje de elevación.<br>n4= valor mínimo del eje de elevación. |

#### 4.2.2. PROGRAMA CONSTELD

Este programa, escrito en Python, se encarga del manejo de las instrucciones de alto nivel. Recibe las mismas por medio del puerto de *socket* TCP/IP número 9999. A su vez, este programa se encarga de generar las instrucciones para el programa **servobb** y de ejecutar rutinas de alto nivel como la búsqueda de inicios.

Las instrucciones del programa **consteld** se listan a continuación.

**TABLA 7**

Instrucciones del programa **consteld**

| Instrucción | Función |
|---|---|
| AHDEC n1 n2 | Mueve el telescopio a las coordenadas n1=(ángulo horario) n2=(declinación).<br>n1 en horas.<br>n2 en grados.<br>Habilita el guiado. |
| ESTADO | Regresa el valor de las coordenadas actuales del telescopio. |
| QUITA_GUIADO | Quita el seguimiento o guiado. |
| LISTO | Quita el guiado. Pone los valores de reposo del telescopio. En general hace un "restablece" para los controladores de motores. |
| AR n [n1] [n2] | Define el valor de la coordenada de ascensión recta deseada. Los parámetros son:<br>• n = horas, puede ser real y especificarse por ejemplo como: 3.4567<br>• n1= minutos, el parámetro es opcional.<br>• n2 = segundos, el parámetro es opcional.<br>Ejemplos:<br>AR 3.4567<br>AR 3 4 5.6 |
| DEC n [n1] [n2] | Coordenada deseada de declinación. Los parámetros son:<br>• n = grados, puede ser real y especificarse por ejemplo como: -3.4567.<br>• n1= minutos, el parámetro es opcional.<br>• n2 = segundos, el parámetro es opcional.<br>Ejemplos:<br>DEC 3.4567<br>DEC 3 4 5.6 |
| EPOCA n | Época de las coordenadas. |
| CORR | Corrige o ajusta el valor de las coordenadas del telescopio a las coordenadas deseadas del último movimiento. |
| LATITUD n | Define la latitud del lugar de observación. Está dado en grados. |
| LONGITUD n | Define la longitud del lugar de observación. Está dado en grados. El valor es negativo al oeste de Greenwich.<br>Ejemplo: LONGITUD -116.11 |
| ACIM n | Mueve el telescopio a la posición de azimut dada por el parámetro. Se da en grados. E.G. : ACIM 190.3 |
| ELEV n | Mueve el eje de elevación del telescopio a la posición dada por el parámetro. Se da en grados. E.G. : ELEV 35.5 |

| Instrucción | Función |
|---|---|
| TEL | Reporta el estado del telescopio. Coordenadas, tiempo universal, tiempo sideral. |
| ACT | Mueve el telescopio a las coordenadas deseadas definidas por las instrucciones AR, DEC, EPOCA. Hace los cálculos de coordenadas aparentes, precesión, nutación, aberración y refracción. |
| ACTP | Mueve el telescopio a las coordenadas deseadas definidas por las instrucciones AR, DEC. No realiza ninguna transformación a las coordenadas. |
| BUSCA_INICIOS | Realiza el proceso de búsqueda de inicios y mueve el telescopio a la posición de reposo. |

#### 4.2.2.1. INSTRUCCIONES DE BAJO NIVEL DEL PROGRAMA CONSTELD

Este conjunto de instrucciones se muestra en la Tabla 8. Se utilizan para definir parámetros de operación del control de motores; el usuario normalmente no utiliza este juego de instrucciones, sólo se hace uso de ellas al momento de caracterizar el movimiento del telescopio.

**TABLA 8**

Instrucciones de bajo nivel del programa **consteld**

| Instrucción | Función |
|---|---|
| OFFINIX n | Valor en pulsos al que se mueve el eje de azimut después de haber concluido la rutina de búsqueda de inicios. Es casi imposible ajustar con una buena precisión los interruptores de inicio de los ejes de azimut y elevación. Para esto son útiles los parámetros OFFINIX y OFFINIY. Esta será la posición definida para el reposo o inicio de operaciones. |
| OFFINIY n | Valor en pulsos al que se mueve el eje de elevación después de haber concluido la rutina de búsqueda de inicios. Ver OFFINIX. |
| REPOSO | Mueve el telescopio a la posición de reposo. Usualmente 180 grados en azimut y 45 grados en el eje de elevación. |
| PPSX n | Define el valor de pulsos por segundo de arco del eje de azimut. |
| PPSY n | Define el valor de pulsos por segundo de arco del eje de elevación. |
| OFFSETA n | Define el desplazamiento que se resta a la posición del eje de azimut. Está en grados. Ver instrucción **SPOFFA**. |
| OFFSETB n | Define el desplazamiento que se resta a la posición del eje de elevación. Está en grados. Ver instrucción **SPOFFA**. |

| Instrucción | Función |
|---|---|
| MAXACIM n | Valor de posición máximo para el eje de azimut que puede alcanzar el telescopio antes de tocar un interruptor límite. Está dado en grados. |
| MINACIM n | Valor de posición mínimo que puede alcanzar el telescopio antes de tocar un interruptor límite. Está dado en grados. |
| MAXELEV n | Valor de posición máximo en el eje de elevación que puede alcanzar el telescopio antes de tocar un interruptor límite. Está dado en grados. |
| MINELEV n | Valor de posición mínimo que puede alcanzar el telescopio antes de tocar un interruptor límite. Está dado en grados. |
| ACTPARS | Manda al programa **servobb** los parámetros actuales del programa **consteld**. |
| LEEARCHCFG | Abre y ejecuta las instrucciones del archivo de configuración "cons.cfg". |
| C2C | Esta es una instrucción especial y se usa para mandar una serie de instrucciones directo al programa **servobb**. Por ejemplo, la secuencia:<br>C2C X= 1000 Y= -10002 FC2C<br>Mueve el eje X a la posición 1000 y el eje Y a la posición -10002. La secuencia debe terminar con la cadena FC2C para ser válida. |

#### 4.3. PROGRAMA DE INTERFAZ DE USUARIO

El programa de interfaz de usuario, **consola.py**, es un programa de interfaz gráfico que se ejecuta en el ambiente de ventanas X de Linux. Fue desarrollado en Python usando los módulos GTK, y GObject, así como la herramienta de desarrollo Glade.

Este programa se ejecuta normalmente en la computadora PC desde la que se opera el telescopio. Su función es generar las instrucciones para la computadora *BeagleBone* del telescopio.

Desde este programa se puede operar el telescopio de manera amigable. Ofrece la capacidad de realizar las siguientes tareas:

- Búsqueda de inicios para empezar la observación
- Mover el telescopio a las coordenadas deseadas
- Definir la posición de reposo
- Movimientos relativos de las coordenadas por medio de botones, simulando una paleta manual

Además, el programa abre un puerto de *socket* TCP/IP para recibir instrucciones de otros procesos. En particular se puede usar para mover el telescopio por medio del programa *Stellarium*, como se describe en la Sección 5.

La *Figura 7* muestra una vista de la ventana principal del programa. El programa **consola.py** acepta los siguientes argumentos o parámetros de la línea de instrucciones:

- [--host HOST]
- [--port PORT]
- --serverport SERVERPORT

El argumento "host" es opcional; define la dirección IP del *BeagleBone* que controla el telescopio. Por omisión toma el valor 192.168.0.205.

El argumento "port" es opcional y define el número el puerto de *socket* del controlador del telescopio. Por omisión toma el valor 9999.

El argumento "serverport" es obligatorio y es el puerto de *socket* donde el programa consola.py recibe las instrucciones de otros procesos.

Las instrucciones que interpreta el programa son las siguientes: AR, DEC, EPOCA, ACT y TEL.

Para información sobre su sintaxis, referirse a la Sección 4.2.2.


***Figura 7:*** *Vista principal del programa de interfaz de usuario.*

## 5. BÚSQUEDA Y APUNTADO DE OBJETOS UTILIZANDO EL PROGRAMA *STELLARIUM*

Se adaptó un programa servidor para el *plugin* de control de telescopio del programa *Stellarium*. El párrafo siguiente se tomó de la página "web" del programa:

> "Stellarium es un programa gratuito de código abierto. Es capaz de mostrar un cielo realista en 3D, tal como se aprecia a simple vista, con binoculares o telescopio".

Por medio de este servidor es posible mover el telescopio desde la ventana gráfica del programa S*tellarium*, lo que permite la búsqueda de objetos desde los catálogos cargados en el programa. Además, permite ver la posición del telescopio en la misma ventana gráfica y desplegar a gusto del usuario toda la información disponible del objeto que se está observando.

Para utilizar el *plugin* de control de telescopio sólo se tiene que ejecutar el programa servidor de tal manera que éste se comunique con el programa de interfaz de consola por medio de su puerto "serverport" (ver la Sección 4.3. donde se especifican los argumentos del programa **consola.py**).

El programa servidor del *plugin* de control del telescopio se llama **TelescopeServerOAN** y se ejecuta desde una terminal de consola del ambiente X de Linux. Los argumentos son posicionales y están definidos por:

1. Argumento 1: Puerto de *socket* del programa *Stellarium*.
2. Argumento 2: Dirección IP de la máquina donde se ejecuta el programa de interfaz gráfica de telescopio **consola.py**.
3. Argumento 3: Puerto de *socket* del programa **consola.py**.

Como ejemplo de la ejecución del programa:

#usuario>./TelescopeServerOAN 9996 localhost 4959

indica que el programa *Stellarium* se conectará al puerto 9996 y el programa de **consola.py** se está ejecutando en esta PC "localhost" y su puerto de *socket* es 4959.

## 6. PRUEBAS Y RESULTADOS

Las pruebas realizadas en el laboratorio fueron de apuntado y seguimiento, para ello se monitoreó la información proporcionada por los codificadores, obteniéndose como resultado errores menores a 1 segundo de arco. Este valor es comparable con lo medido en el telescopio de 84cm del OAN SPM.

## 7. CONCLUSIONES

Se presentó el diseño de un sistema de control de tiempo real para un telescopio con montura del tipo elevación–azimut basado en una computadora de una sola tarjeta modelo *BeagleBone Black*.

Se desarrollaron las tarjetas electrónicas para la interfaz *BeagleBone* motor-codificador telescopio.

Se desarrollaron los algoritmos de control, programas de tiempo real, programas servidores y la interfaz de usuario.

También se implementó la interfaz al programa *Stellarium*.

Las pruebas de movimiento y el desempeño logrado con los algoritmos de control indican la viabilidad del uso de un *BeagleBone* para el control de los telescopios del OAN.

## 8. REFERENCIAS

[1] http://beagleboard.org/black

[2] Zazueta, S.
"La Biblioteca COMOAN 0.1.0"
*Publicaciones Técnicas del Instituto de Astronomía, UNAM.*
Reporte Técnico. RT-2009-05.
México, 2009.

[3] Zazueta, S., Murillo, F.
"Manual De Usuario: Instrucciones del Programa de Control del Telescopio de 0.84m del OAN, Versión 1.3-L",
*Publicaciones Técnicas del Instituto de Astronomía, UNAM.*
Manual de Usuario. MU-2001-01.
México, 2001.

[4] Ibarra, R., Zazueta, S.
"Modificaciones a la Tarjeta de Interfaz del Telescopio de 1.5 Metros. Avances en el Diseño",
*Publicaciones Técnicas del Instituto de Astronomía, UNAM.*
Reporte Técnico #100.
México, 1992.

[5] Zazueta S.
"Manual del Programa de Consola del Telescopio de 1.5 Metros"
*Publicaciones Técnicas del Instituto de Astronomía, UNAM.*
Reporte Técnico #81.
México, 1990.

[6] Documentación de la biblioteca GLIB.
https://developer.gnome.org/glib/

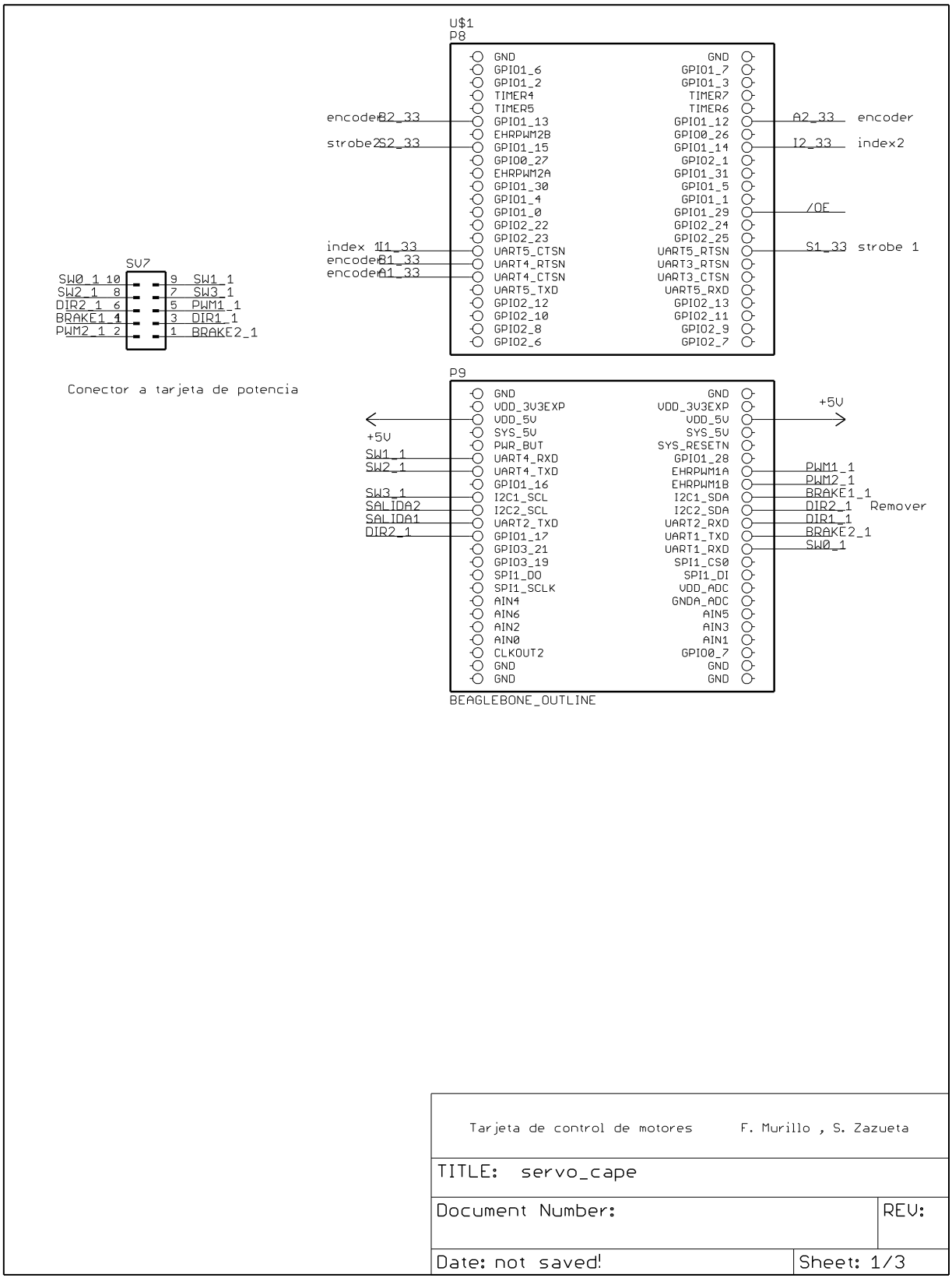
[7] Documentación de la biblioteca GTK+.
http://www.gtk.org/

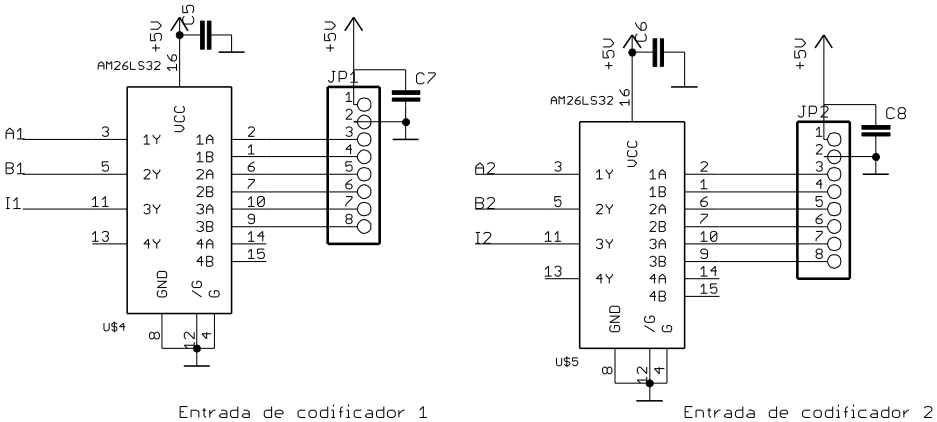
[8] Documentación BeagleBone Black.
http://beagleboard.org/black

[9] Documentación Xenomai Real Time Linux.
http://xenomai.org/

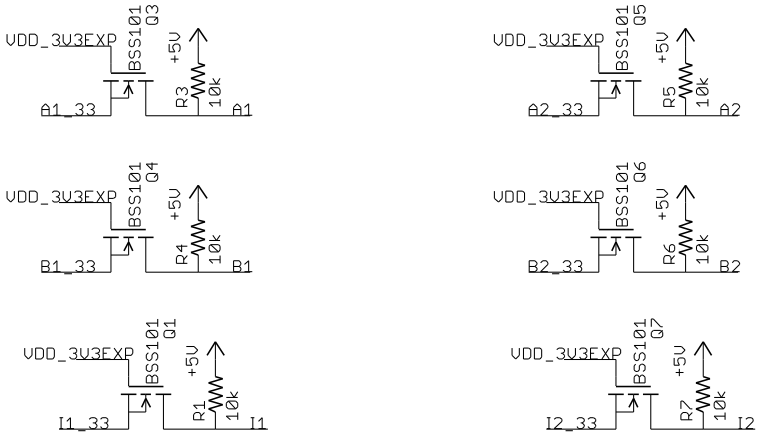
[10] Documentación del procesador AM335x Sitara ARM Cortex-A8
http://www.ti.com/lit/pdf/spruh73

# APÉNDICE A. DIAGRAMA ESQUEMÁTICO Y MAPA DE COMPONENTES DE LA TARJETA DE INTERFAZ



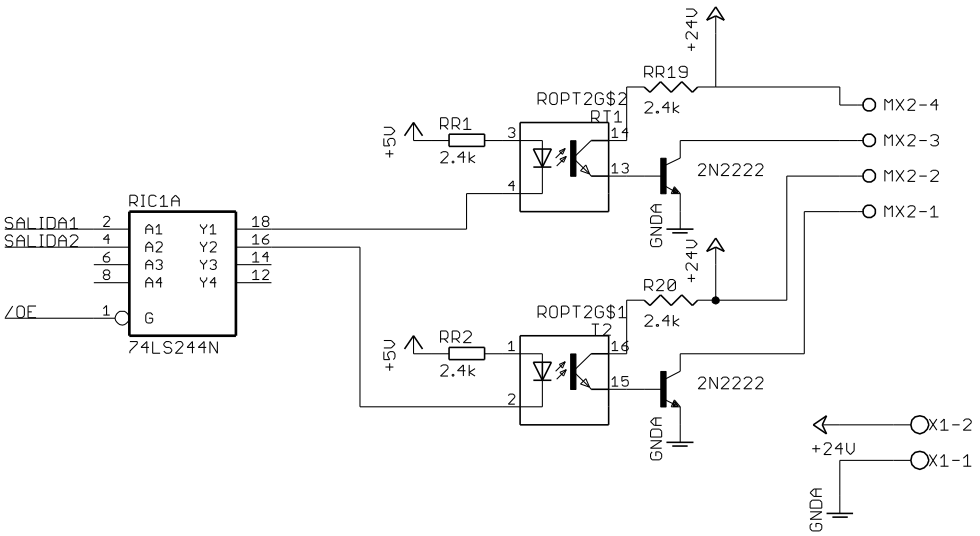



Entrada de codificador 1 Entrada de codificador 2

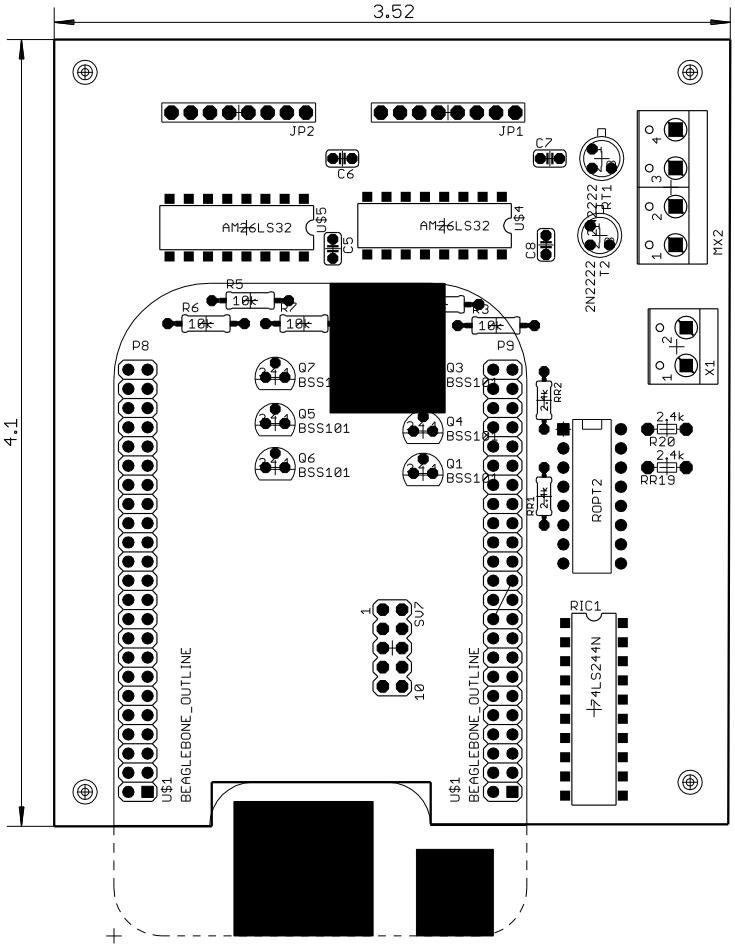



Cambiadores de nivel 5V a 3.3V

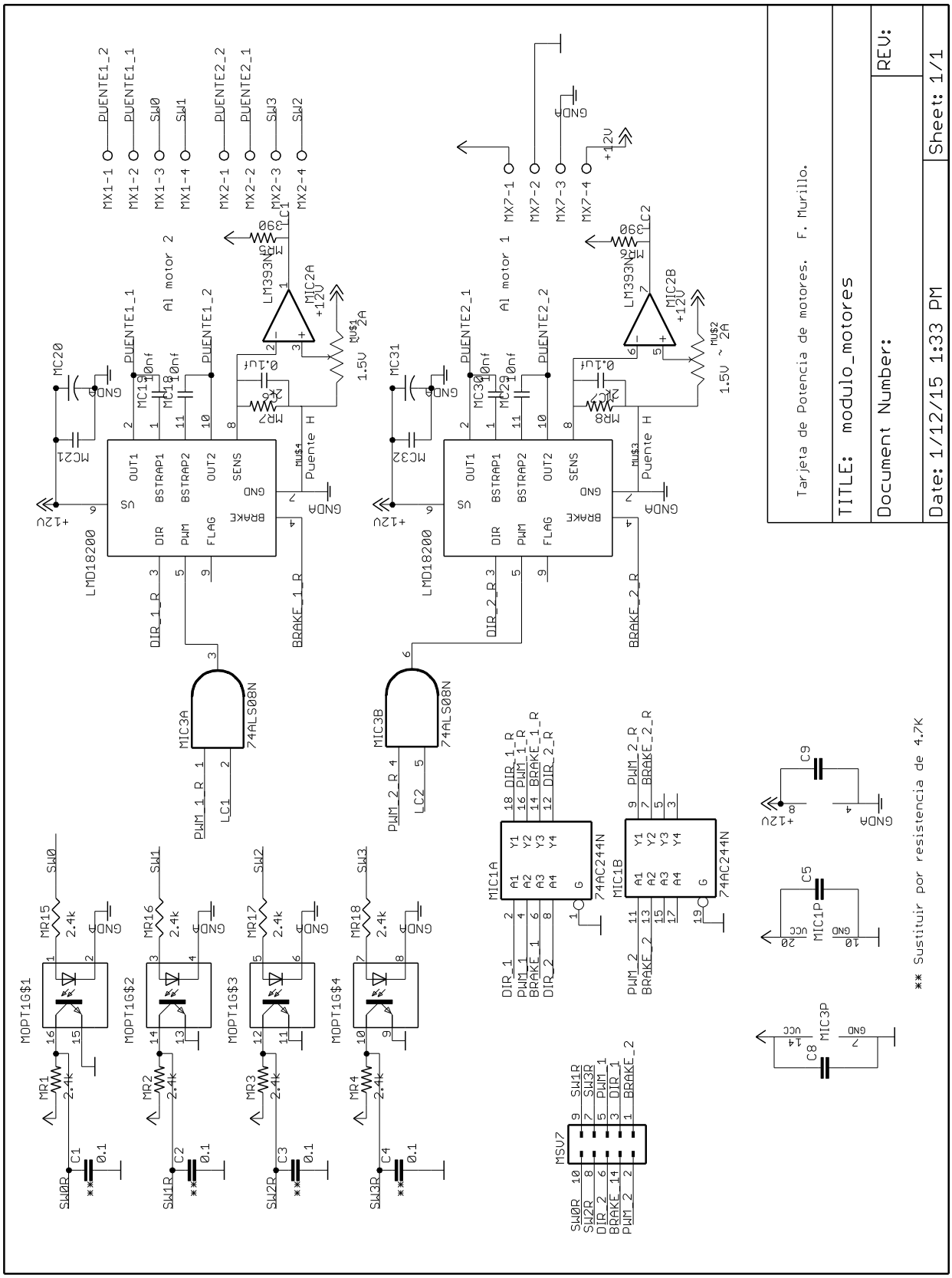
| | |
|---|---|
| Tarjeta de control de motores F. Murillo , S. Zazueta | |
| TITLE: servo_cape | |
| Document Number: | REV: |
| Date: not saved! | Sheet: 2/3 |

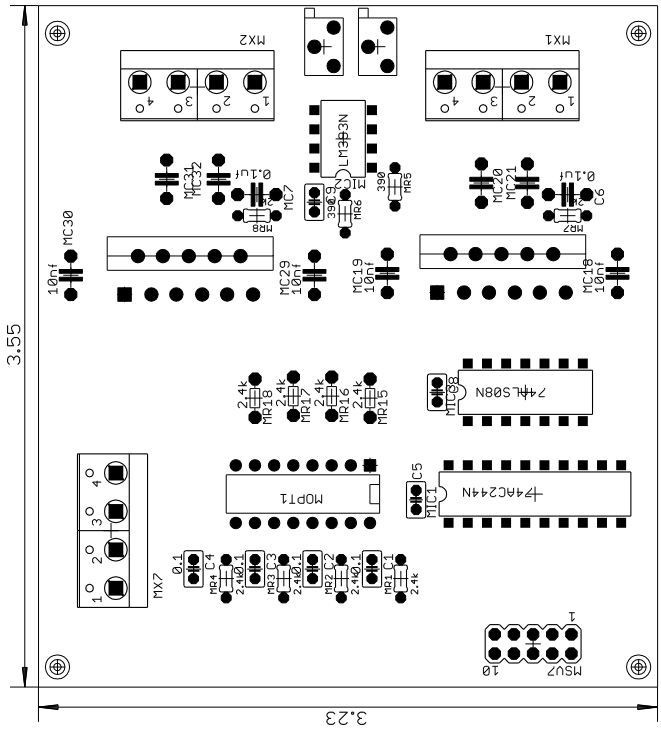



| | | |
|---|---|---|
| Tarjeta de control de motores F. Murillo , S. Zazueta | | |
| TITLE: servo_cape | | |
| Document Number: | | REV: |
| Date: not saved! | Sheet: 3/3 | |

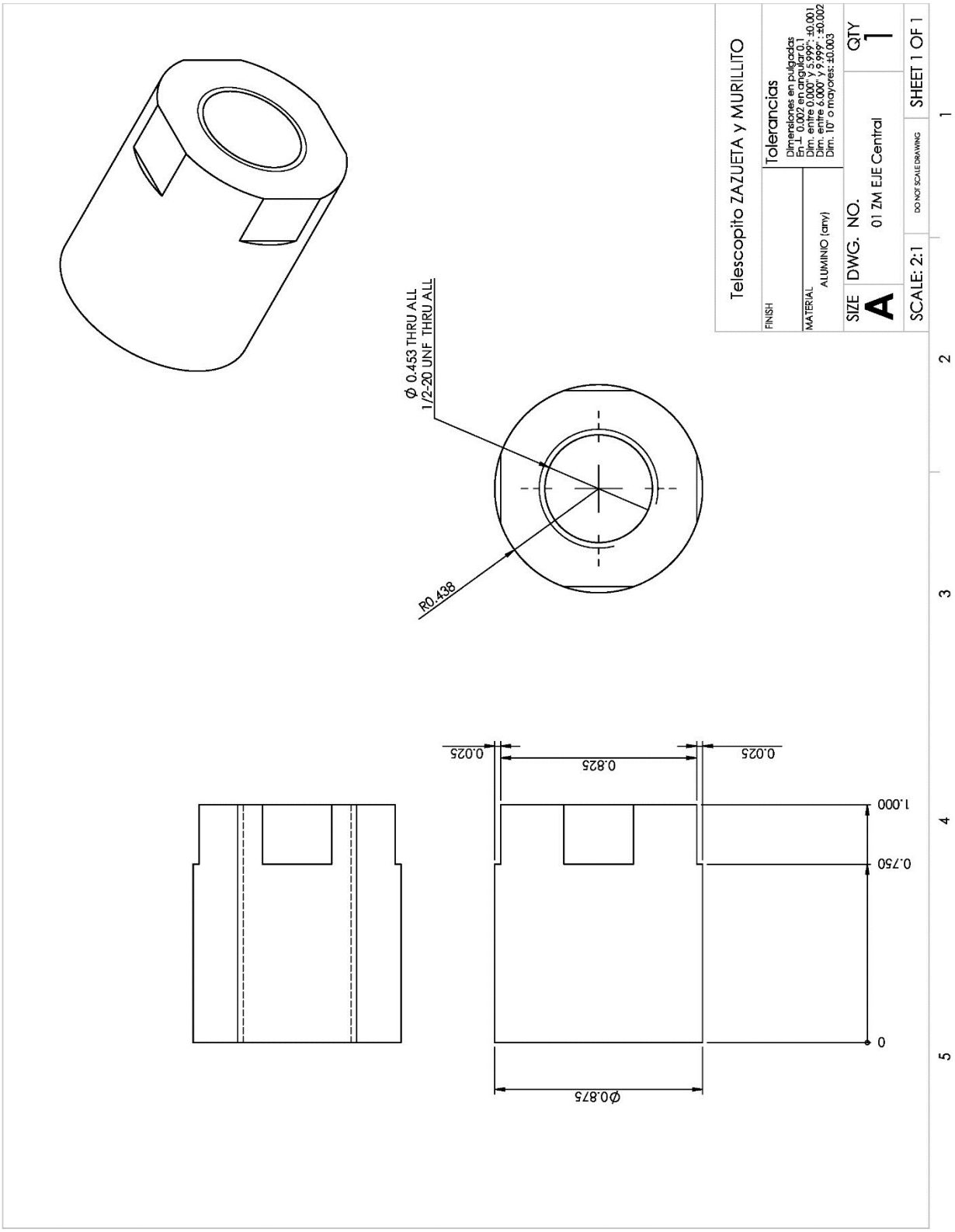

3.52
4.1
JP2
JP1
C7
C6
C5
C8
U$5
U$4
AM26LS32
AM26LS32
2N2222
2N2222
T1
T2
R11
MX2
R5
R6
R7
R3
10k
P8
P9
Q7
BSS1
Q3
BSS1
Q5
BSS101
Q4
BSS1
Q6
BSS101
Q1
BSS1
RR2
RR1
2.4k
R20
RR19
ROPT2
X1
SV7
10
RIC1
74LS244N
U$1
BEAGLEBONE_OUTLINE
U$1
BEAGLEBONE_OUTLINE

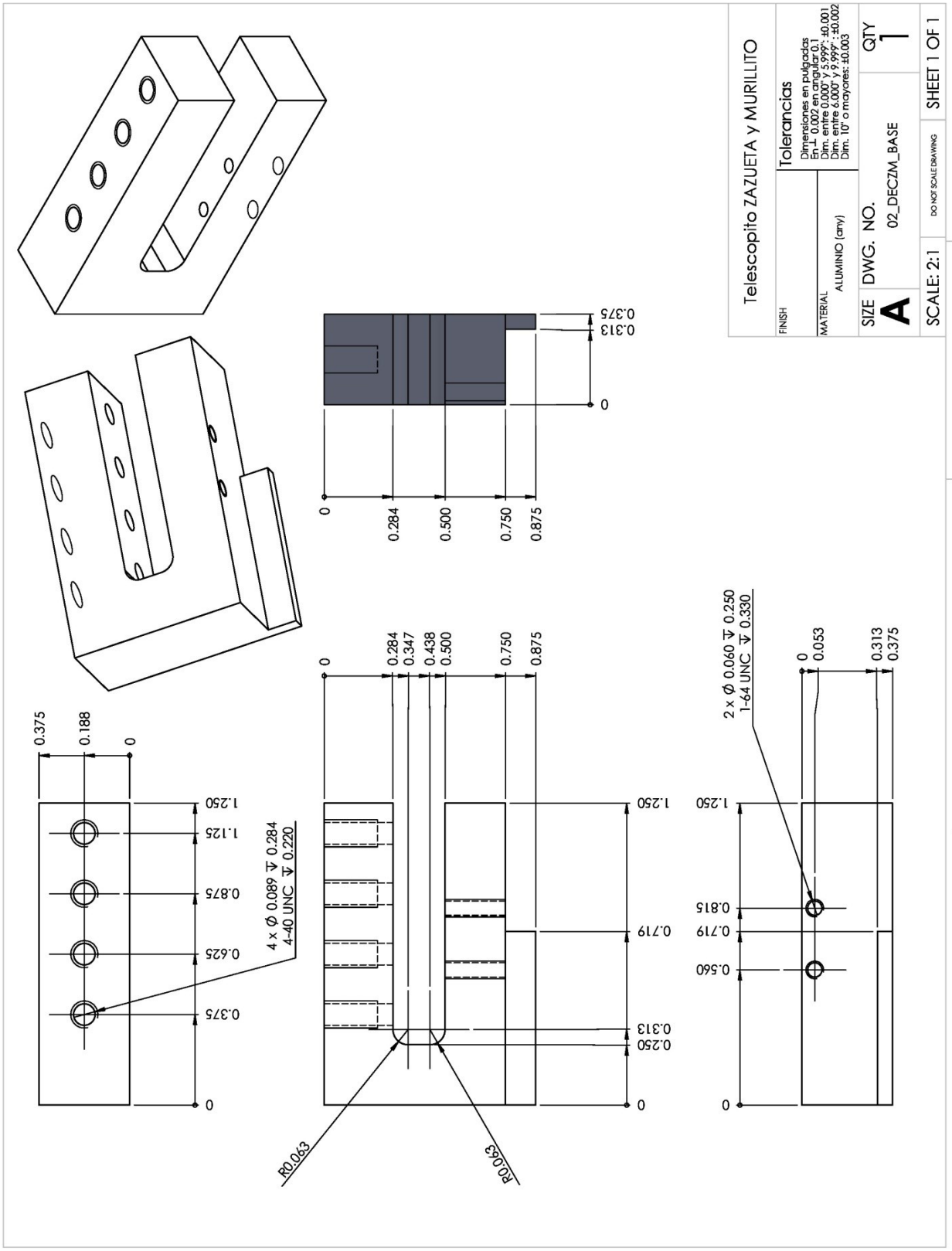
## APÉNDICE B. DIAGRAMA ESQUEMÁTICO Y MAPA DE COMPONENTES DE LA TARJETA DE POTENCIA DE MOTORES

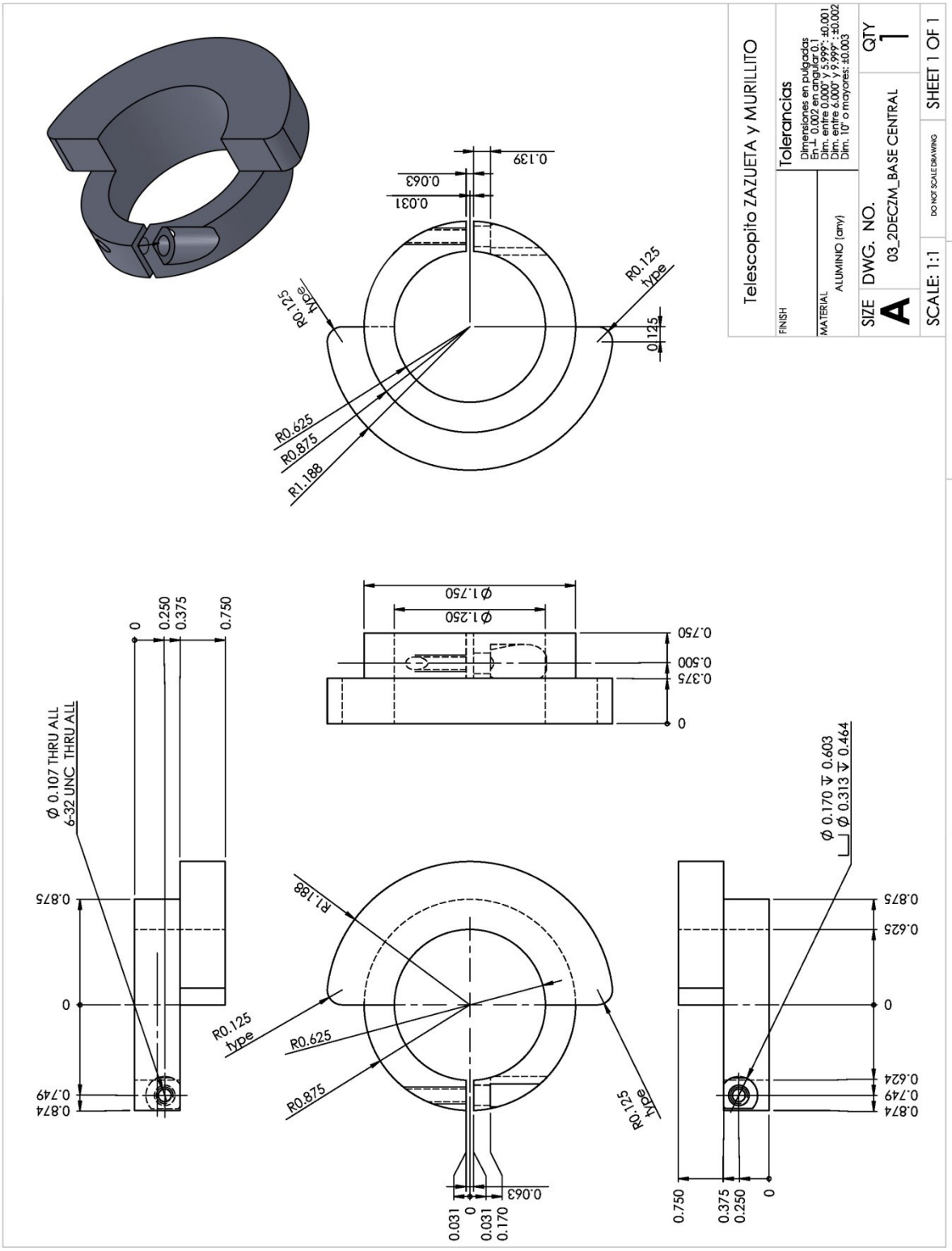


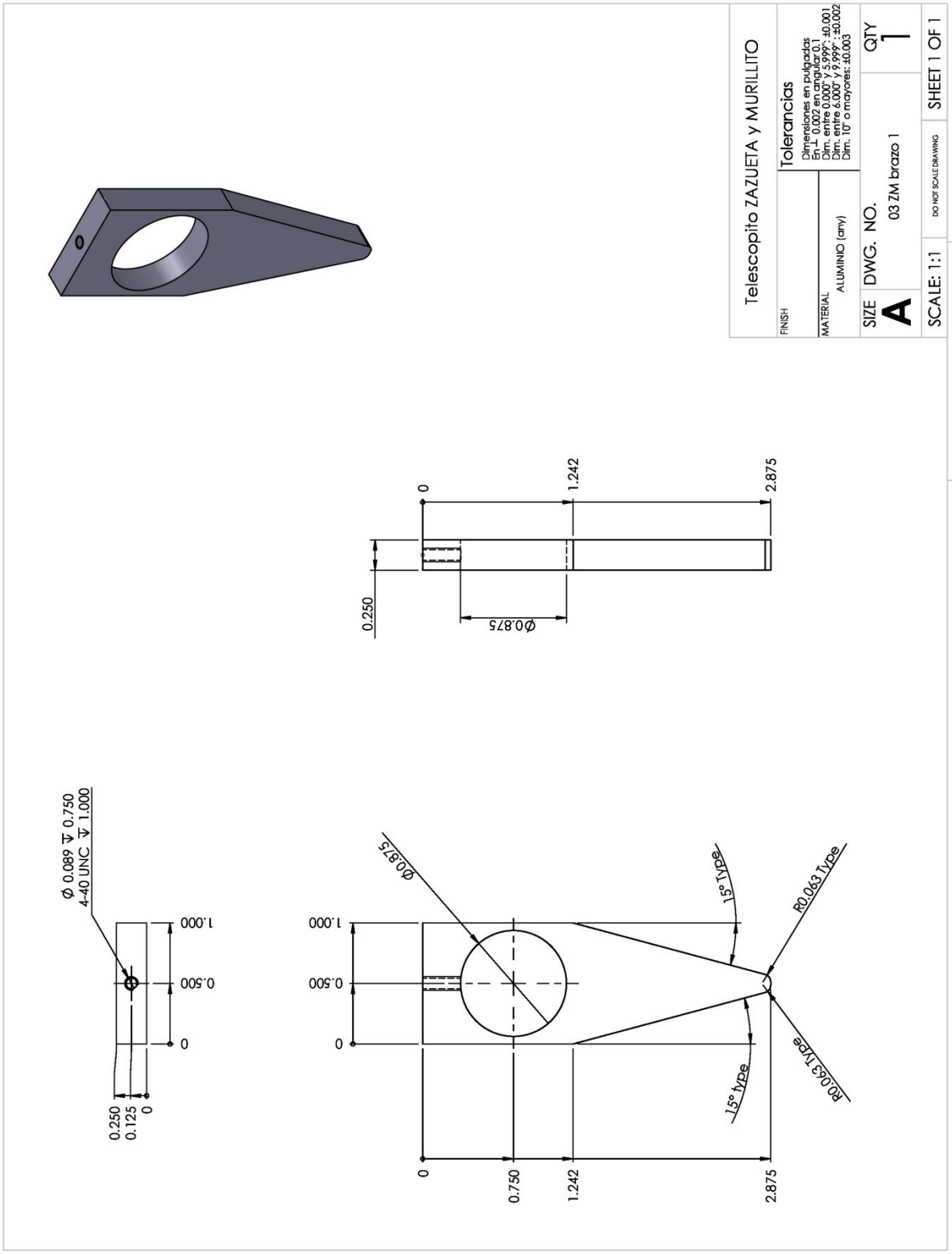

3.55
3.23
MX2
MX1
LM386N
MC31
MC32
MC7
MC20
MC21
MC6
MC30
MC29
MC19
MC18
MR18
MR17
MR16
MR15
MOPT1
74AC244N
MC1
MC5
MX7
MR4 C4
MR3 C3
MR2 C2
MR1 C1
MSV7

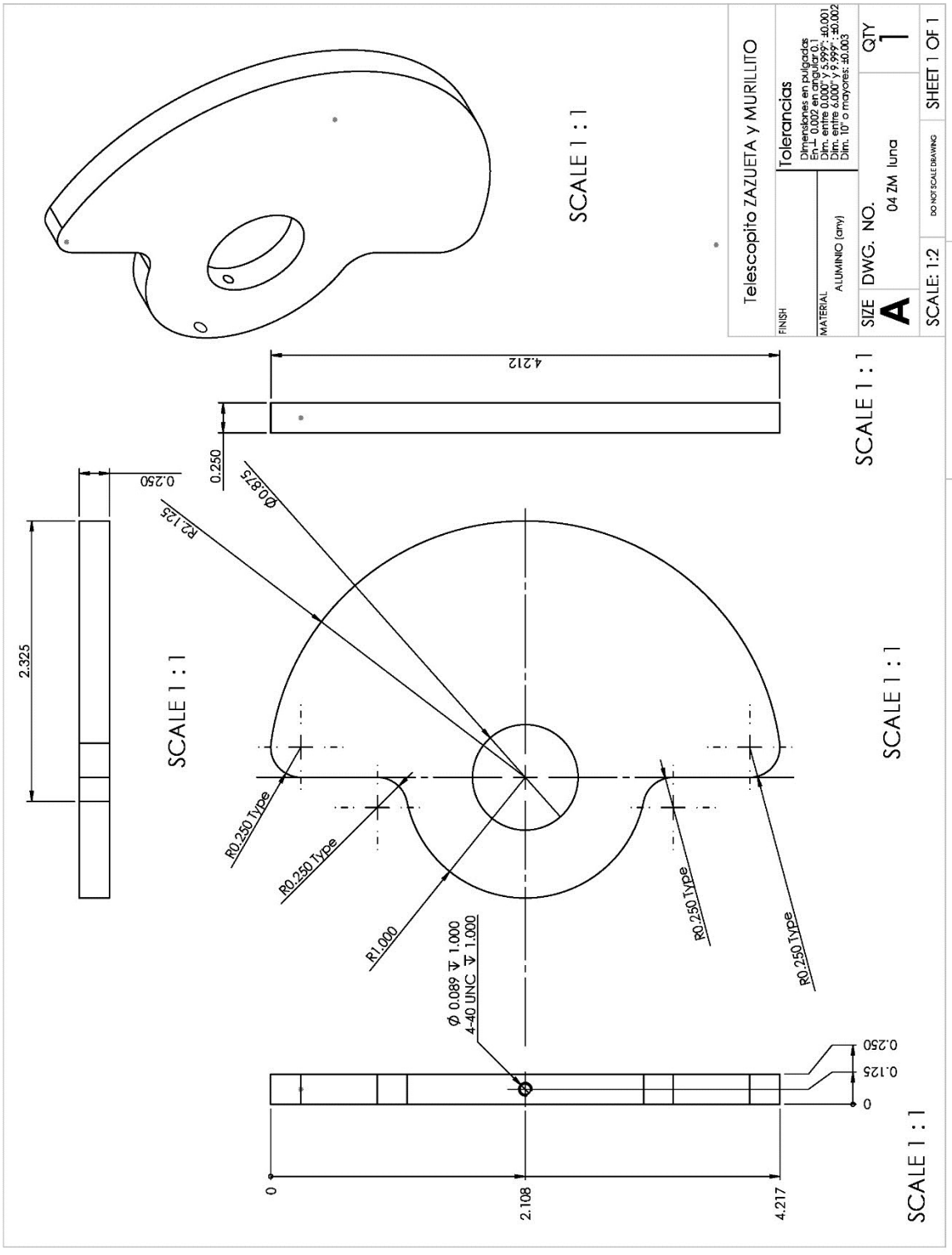
## APÉNDICE C. DIAGRAMAS MECÁNICOS

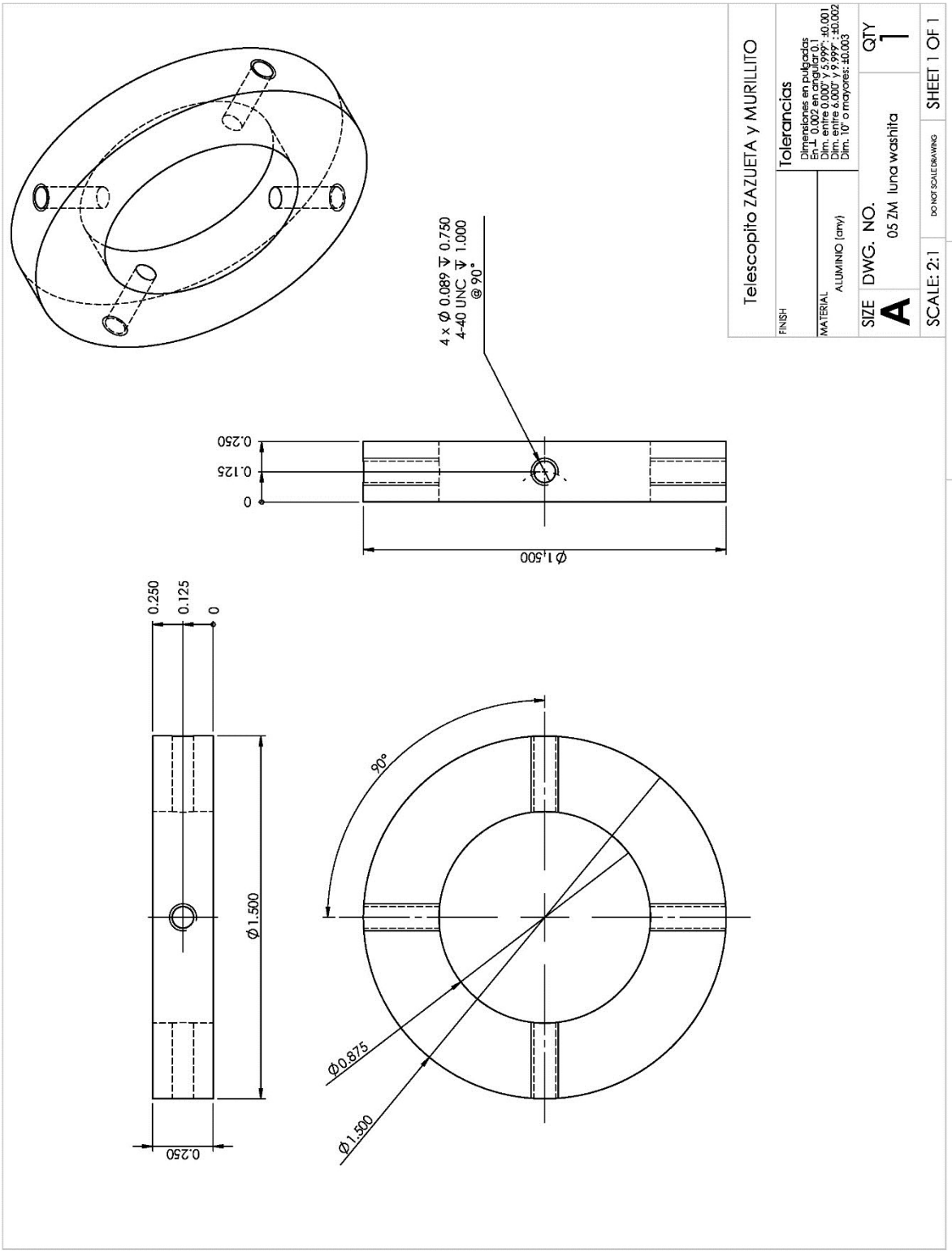


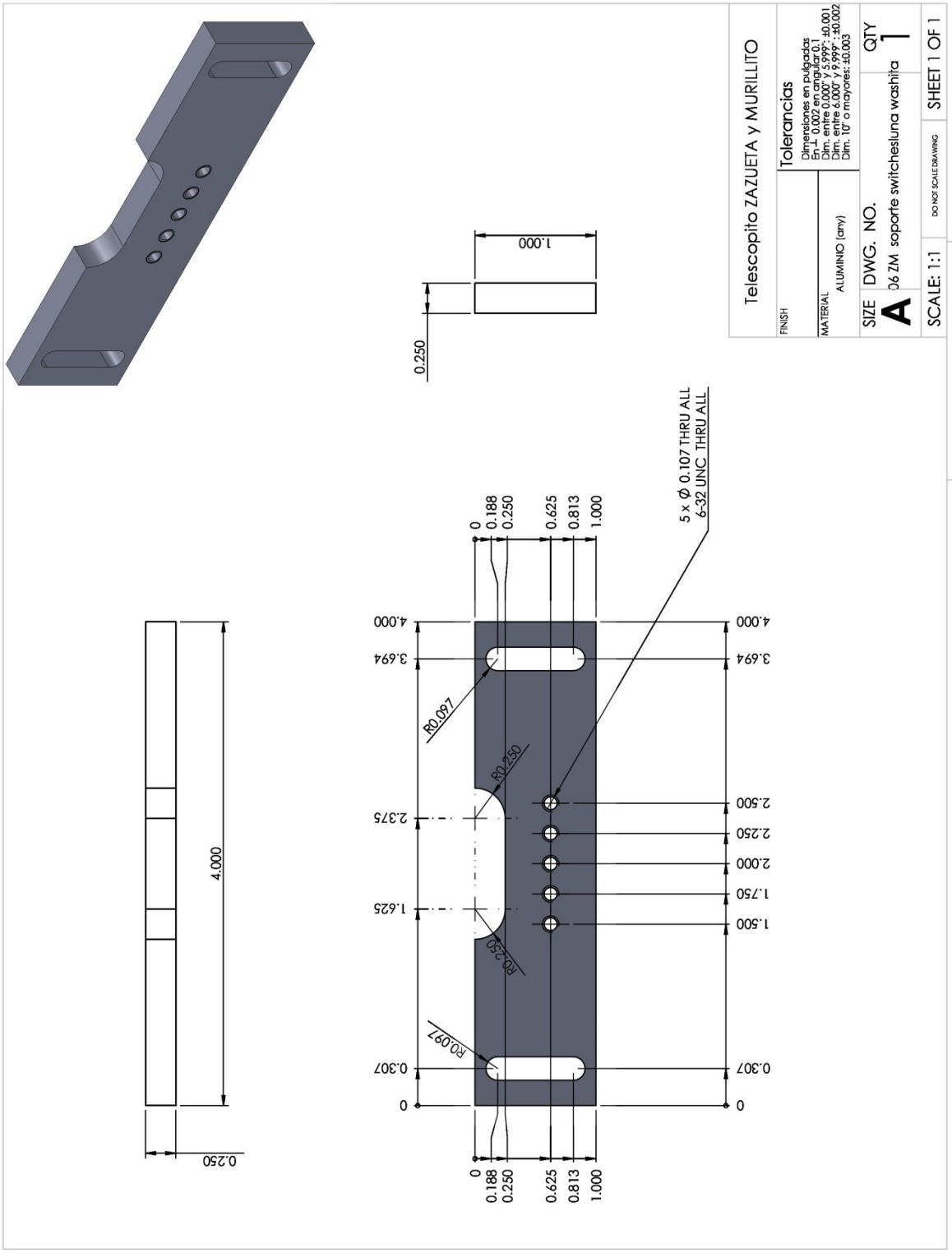

Telescopito ZAZUETA y MURILLITO
FINISH
Tolerancias
Dimensiones en pulgadas
En ± 0.002 en angular 0.1
Dim. entre 0.000" y 5.999": ±0.001
Dim. entre 6.000" y 9.999": ±0.002
Dim. 10" o mayores: ±0.003
MATERIAL
ALUMINIO (amy)
SIZE
A
DWG. NO.
02_DEC7M_BASE
QTY
1
SCALE: 2:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
0.375
0.313
0
0.284
0.500
0.750
0.875
0.188
1.250
1.125
0.875
0.625
0.375
4 x Ø 0.089 ∓ 0.284
4-40 UNC ∓ 0.220
0.347
0.438
0.719
0.250
R0.063
2 x Ø 0.060 ∓ 0.250
1-64 UNC ∓ 0.330
0.053
0.815
0.560
1
2
3
4
5

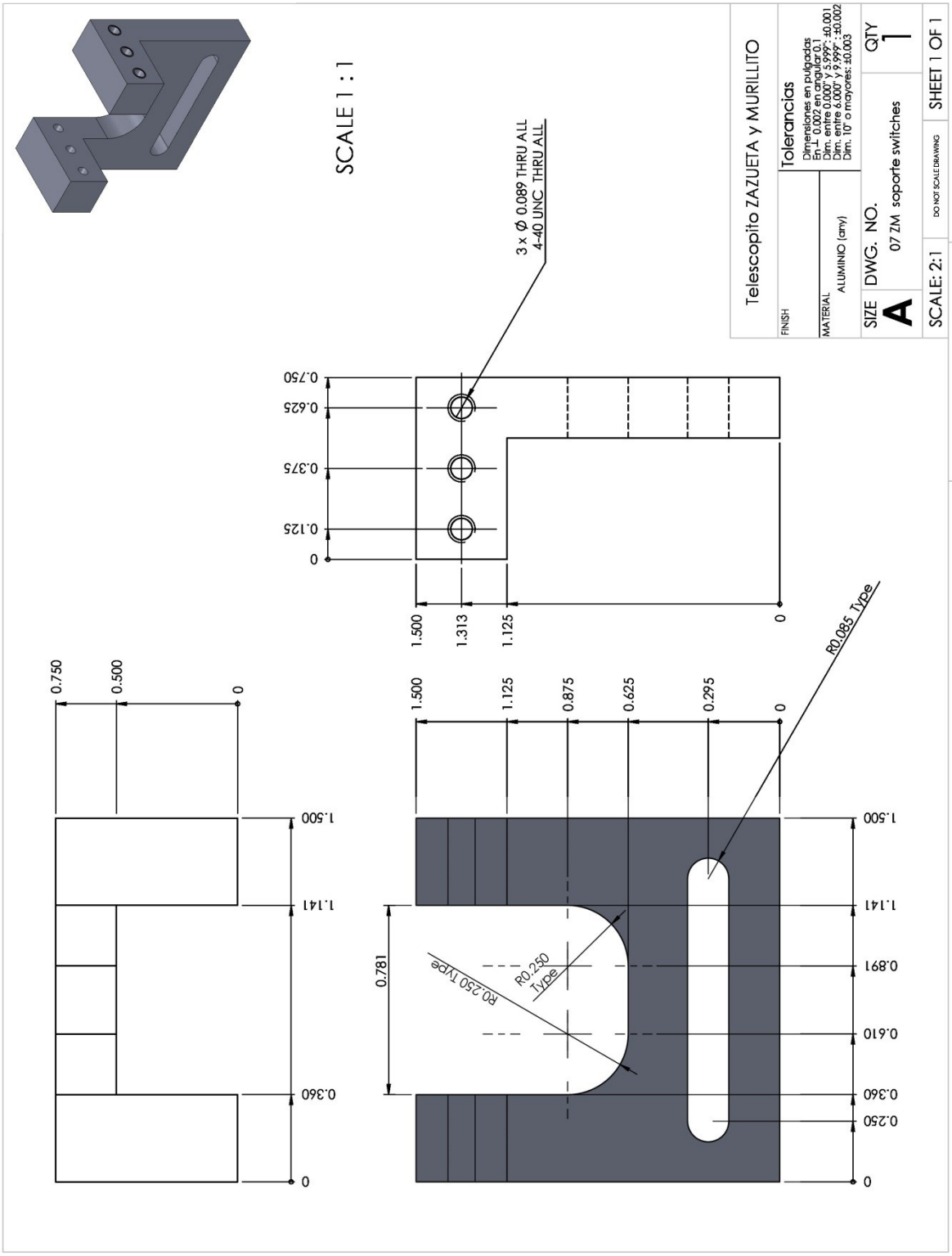

Telescopito ZAZUETA y MURILLITO
Tolerancias
Dimensiones en pulgadas
En ⊥ 0.002 en angular 0.1
Dim. entre 0.000" y 5.999": ±0.001
Dim. entre 6.000" y 9.999": ±0.002
Dim. 10° o mayores: ±0.003
FINISH
MATERIAL
ALUMINIO (amy)
SIZE
A
DWG. NO.
03_2DECZM_BASE CENTRAL
QTY
1
SCALE: 1:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
0.139
0.063
0.031
R0.125 type
0.125
R0.625
R0.875
R1.188
Ø 1.750
Ø 1.250
0.750
0.500
0.375
0
Ø 0.107 THRU ALL
6-32 UNC THRU ALL
0.250
0.375
0.750
0.875
0.874
0.749
Ø 0.170 ↧ 0.603
⌴ Ø 0.313 ↧ 0.464
0.625
0.624
0.031
0.170


Telescopito ZAZUETA y MURILLITO
FINISH
Tolerancias
Dimensiones en pulgadas
En 1 Cifra: ±0.1
En 2 Cifras: ±0.01
Dim. entre 0.000" y 5.999": ±0.001
Dim. entre 6.000" y 9.999": ±0.002
Dim. 10" o mayores: ±0.003
MATERIAL
ALUMINIO (amy)
SIZE
A
DWG. NO.
03 ZM brazo 1
QTY
1
SCALE: 1:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
0
1.242
2.875
0.250
Ø0.875
Ø 0.089 ↧ 0.750
4-40 UNC ↧ 1.000
1.000
0.500
0
0.250
0.125
0
15° Type
R0.063 Type
0.750


Telescopito ZAZUETA y MURILLITO
Tolerancias
Dimensiones en pulgadas
En .1 0.002 en angular 0.1
Dim. entre 0.000" y 5.999": ±0.001
Dim. entre 6.000" y 9.999": ±0.002
Dim. 10° o mayores: ±0.003
FINISH
MATERIAL
ALUMINIO (any)
SIZE A
DWG. NO. 04 ZM luna
QTY 1
SCALE: 1:2
DO NOT SCALE DRAWING
SHEET 1 OF 1
SCALE 1 : 1
4.212
0.250
Ø0.875
R2.125
0.250
2.325
R0.250 Type
R0.250 Type
R1.000
Ø 0.089 ∓ 1.000
4-40 UNC ∓ 1.000
R0.250 Type
R0.250 Type
0.250
0.125
0
0
2.108
4.217


Telescopito ZAZUETA y MURILLITO
FINISH
Tolerancias
Dimensiones en pulgadas
En ⊥ 0.002 en angulares 0.1
Dim. entre 0.000" y 5.999" : ±0.001
Dim. entre 6.000" y 9.999" : ±0.002
Dim. 10" o mayores: ±0.003
MATERIAL
ALUMINIO (any)
SIZE
A
DWG. NO.
05 ZM luna washita
QTY
1
SCALE: 2:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
4 x Ø 0.089 Ŧ 0.750
4-40 UNC Ŧ 1.000
@ 90°
0.250
0.125
0
Ø1.500
90°
Ø0.875
Ø1.500
0.250
0.125
0
Ø1.500
0.250


Telescopito ZAZUETA y MURILLITO
Tolerancias
Dimensiones en pulgadas
En 1 0.002"en angular 0.1
Dim. entre 0.000" y 5.999" : ±0.001
Dim. entre 6.000" y 9.999" : ±0.002
Dim. 10" o mayores: ±0.003
FINISH
MATERIAL
ALUMINIO (any)
SIZE
A
DWG. NO.
06 ZM soporte switchesluna washita
QTY
1
SCALE: 1:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
1.000
0.250
5 x Ø 0.107 THRU ALL
6-32 UNC THRU ALL
0
0.188
0.250
0.625
0.813
1.000
4.000
3.694
2.375
1.625
0.307
2.500
2.250
2.000
1.750
1.500
R0.097
R0.250
4.000
0.250


SCALE 1 : 1
3 x Ø 0.089 THRU ALL
4-40 UNC THRU ALL
Telescopito ZAZUETA y MURILLITO
Tolerancias
Dimensiones en pulgadas
En ⊥ 0.002 en angular 0.1
Dim. entre 0.000" y 5.999": ±0.001
Dim. entre 6.000" y 9.999": ±0.002
Dim. 10° o mayores: ±0.003
FINISH
MATERIAL
ALUMINIO (amy)
SIZE
A
DWG. NO.
07 ZM soporte switches
QTY
1
SCALE: 2:1
DO NOT SCALE DRAWING
SHEET 1 OF 1
R0.085 Type
R0.250 Type
R0.250 Type
0.781