Proyecto final. Sistemas Electrónicos Lineales. Sistema de alarma para una motocicleta que funge como repartidor.

Marcoc-rasi

OBJETIVO

General

- Brindar un sistema que de seguridad a la motocicleta de las personas que trabajan en el área de repartidores de alimentos y/o objetos.

- El objetivo del curso es brindar a los estudiantes conocimientos y bases teóricas sólidas y permanentes y no solo conocimientos temporales que el estudiante use para acreditar la evaluación del contenido de la unidad o tema del programa y posteriormente olvidarlo.

1. Introducción

El proyecto por desarrollar busca la protección de diversas áreas de una motoneta que es utilizada para repartir paquetes, el usuario que opera dicho vehículo ha señalado muchos puntos que pueden ser mejorados con una tecnología vista en el transcurso de la materia.

Como primer factor, todos los motores pueden sufrir sobrecalentamiento, por múltiples causas, este calentamiento es especialmente riesgoso en la culata (cabeza del motor) donde se encuentran las mayores temperaturas de trabajo por lo que implementar un sensor que de aviso cuando se llegue a cierta temperatura es lo planeado.

Por otra parte, la inseguridad hoy día es un hecho que preocupa a cada persona, la motoneta actualmente cuenta con dos sistemas de almacenamiento, la cajuela que se encuentra debajo del asiento, y la caja, por lo que en caso de profanar las pertenencias se deberá de abrir la cajuela o la caja, por lo que tendrá una entrada más fuerte de luz, de esta forma se puede saber si alguien tuvo acceso a las cosas usando un sensor y activando una alarma. De igual forma se propone implementar un sistema de seguridad que pida un código para lograr accesar a los dos lugares para almacenar cosas y otro sensor que active las alarmas si alguien ajeno al conductor designado se sienta en el asiento del vehículo en cuestión. Se usará el servomotor para abrir la cajuela.

Se usará un acelerómetro para detectar el movimiento de la moto, este también activará la alarma. El MOC y el Triac se usarán para encender y apagar las luces de la moto de forma intermitente cuando la alarma se dispare.

Por último, se ve la opción de encender la motoneta de forma automática, esto se logra de modo normal a través de un botón en la moto que manda una señal al motor de arranque eléctrico que hace girar el cigüeñal para que este logre encender, este encendido se puede realizar automáticamente con el sistema que ya tiene la moto y se propone realizar esta acción desde un control sin la necesidad de hacerlo manualmente al usar un sensor infrarrojo para encender y apagar la moto cuando sea necesario a través de un botón.

El teclado matricial es utilizado para colocar una contraseña para que se pueda mover el servomotor y así abrir la cajuela o la caja. La fotorresistencia es utilizada para modular la frecuencia a la cual las luces parpadean. Por último, el display es utilizado para desplegar un mensaje que te indica que la alarma está funcionando. Si el pin Pb1 va a 0, se despliega un mensaje que pide el código para abrir la cajuela, si este se va a 1, se enciende la alarma indicando que estas protegido. Si se coloca la contraseña correctamente, gira a la derecha, si no gira a la izquierda.

- Materiales varios

-Cables
-Resistencias
-Transistor 2n2222
-Foco de 30 Watts
-Socket para foco

-Clavija
-Protoboard

2. Metodología y Materiales

●Para controlar todos los dispositivos y sensores se utilizará el microcontrolador Tiva C Series TM4C1294NCPDT

Figura 1. Tiva C Series TM4C129NCPDT

● Para la medición de la temperatura de la culata del motor se propone implementar un LM35.

Figura 2. Sensor de temperatura LM35 con su encapsulado.

El sensor LM35 requiere de su correcta calibración para asegurar un funcionamiento eficaz, para ello se colocó el pin VCC a 5 VDC, el GND a GND de la fuente de alimentación y al negativo de un multímetro que nos ayudará con dicha tarea y Vsalida estará conectado al positivo del multímetro.

Basándose en los datos obtenidos en el datasheet del sensor se sabe que tiene una pendiente de 10 mV / Cº por lo que si el multímetro indica una medida de 250 mV se sabrá que la temperatura que está detectando el sensor es de 25ºC.

Debido a que la culata del motor llega a temperaturas de 200ºC se simulará con el sensor LM35 que al marcar 40ºC será el equivalente a los 200º reales por lo que tendrá una relación 1:5 para esta aplicación.

● Fotorresistencia

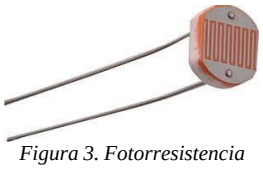

Figura 3. Fotorresistencia

Es una resistencia la cual varía su valor en función de la cantidad de luz que incide sobre su superficie. Cuanto mayor sea la intensidad de luz que incide en la superficie del LDR o fotoresistor menor será su resistencia y en cuanto menor sea la luz que incida sobre este mayor será su resistencia. Cuando la fotoresistencia no está expuesta a radiaciones luminosas, los electrones están firmemente unidos en los átomos que lo conforman, pero cuando sobre ella inciden radiaciones luminosas, esta energía libera electrones con lo cual el material se hace más conductor, y de esta manera disminuye su resistencia. Estas resistencias solamente reducen su resistencia con una radiación luminosa situada dentro de una determinada banda de longitudes de onda. El fotoresistor construido con sulfuro de cadmio son sensibles a todas las radiaciones luminosas visibles. Pueden llegar a medir en la oscuridad valores cercanos a 1 MΩ y expuestas a la luz mediremos valores alrededor de los 100 Ω.

● Optoacoplador MOC 3021 para acoplar el microcontrolador al foco de alarma.

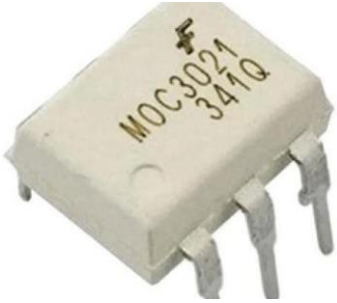

Figura 4. Optoacoplador MOC 3021

Es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente opto electrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo semiconductor, un foto emisor y un foto receptora cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP. El MOC3021-M consta de un diodo emisor de infrarrojos de arseniuro de galio ópticamente acoplado a un interruptor bilateral de silicio. Este dispositivo está diseñado para su uso en aplicaciones que requieren disparo aislado de TRIAC Sus especificaciones son:

-Peso: 9 g
-Tensión máxima de bloqueo de 400 V
-Tensión de aislamiento de 4.17 KV
-Modo de funcionamiento triac sin cruce de cero

-Controlador de salida Diseñado para la línea de 240 VAC
-La corriente del disparador LED es 15mA
-Voltaje de aislamiento: 5.3 kV
-Temperatura de operación: -40 – 85°C
-Corriente continua If: 60 mA

● Un servomotor para abrir las cerraduras de la caja y la cajuela del asiento. Se utilizará un servo de rotación posicional, este es el tipo más común de servomotor. El eje de salida gira aproximadamente la mitad de un círculo, o 180 grados. Tiene topes físicos colocados en el mecanismo de engranaje para evitar que se gire más allá de estos límites para proteger el sensor de rotación.

Dicho servo se utilizará para que al recibir la señal del teclado 4x4 con el código correcto, se abra la caja y/o la cajuela del asiento y así solo el operario del vehículo tendrá acceso a ese compartimiento.

Figura 5. Servomotor comercial SG90

El motor en el interior de un servomotor es un motor DC común y corriente. El eje del motor se acopla a una caja de engranajes similar a una transmisión. El circuito electrónico es el encargado de manejar el movimiento y la posición del motor. El sistema de engranajes hace que cuando movemos el eje motor se sienta una inercia muy superior a la de un motor común y corriente. Los servomotores poseen tres cables. Se necesita una señal de control modulada para poder utilizarlo, haciendo uso de modulación por ancho de pulsos (PWM). El eje del motor DC está acoplado a un potenciómetro, el cual permite formar un divisor de voltaje. El voltaje de la salida del divisor varía en función de la posición del eje del motor DC. Este servomotor cuenta con las siguientes características:

-Peso: 9 g
-Dimensiones: 22.2 x 11.8 x 31mm
-Momento de torsión: 1.8 kgf *cm
-Voltaje de operación: 4.8V
-Ancho de banda muerta: 1µs
-Velocidad de operación: 0.1s/60°
-Temperatura de operación: 0-55 °C
-Ciclo de pulso: 20 ms
-Ancho de pulso: 500-2400 µs

● Para detectar movimiento dentro de la posición del conductor se utilizará un hc-sr501.

Figura 6. Sensor de movimiento hs-sr501

El Sensor PIR HC-SR501 ayuda a captar movimiento, casi siempre se utiliza para detectar si un ser humano se ha movido dentro o fuera de la gama del sensor, en este caso se utilizará para detectar si alguna persona ajena al dueño del vehículo se posiciona dentro del área del conductor.
Es pequeño, de bajo costo, bajo consumo de energía ademas de ser fácil de usar. El HC-S501 incluye un retardo ajustable antes de disparar (aproximadamente 0,5 a 200 segundos) y tiene una sensibilidad ajustable y dos orificios de montaje M2. Su rango de detección es de hasta 7 metros con un angulo de 120 grados. Dicho sensor cuenta con las siguientes características.

-Voltaje de operación: 5v-20v
-Rango de detección: 3-7 metros (ajustable mediante trimmer SX)
-Lente fresnel de 19 zonas, ángulo <120º
-Salida activa alta a 3.3 V
-Tiempo en estado activo de la salida ajustable mediante trimmer (Tx)
-Redisparo configurable mediante jumper de soldadura
-Temperatura de trabajo: -15°C-70°C

● Teclado matricial 4x4 para establecer una contraseña para abrir la caja y la cajuela y en caso de robo apagar el motor y colocar una contraseña de reinicio de encendido.

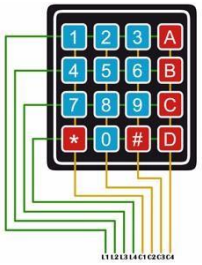

Figura 7. Teclado 4x4 para abrir o cerrar la cajuela

El teclado matricial 4x4 está formado por una matriz de pulsadores dispuestos en filas (L1, L2, L3, L4) y columnas (C1, C2, C3, C4), con la intención de reducir el número de

pines necesarios para su conexión. Las 16 teclas necesitan sólo 8 pines del microcontrolador en lugar de los 16 pines que se requerirían para la conexión de 16 teclas independientes. Para poder leer que tecla ha sido pulsada se debe de utilizar una técnica de barrido y no solo leer un pin de microcontrolador.

El teclado es de tipo membrana, por lo que entre sus ventajas se encuentra el poco espacio que requiere para ser instalado y se vuelve la mejor opción para el limitado espacio donde se colocalará y entra dentro del rango de las temperaturas a las que será sometido.

Sus especificaciones técnicas son las siguientes:

-16 botones con organización matricial (4 filas x 4 columnas)
-Teclado tipo membrana
-Mayor resistencia al agua y al polvo
-Auto adhesivo en la parte de atrás
-Tiempo de rebote (Bounce time): ≤5 ms
-Máximo voltaje operativo: 24 V DC
-Máxima corriente operativa: 30 mA
-Resistencia de aislamiento: 100 MΩ (@ 100 V)
-Voltaje que soporta el dieléctrico: 250 VRMS (@ 60Hz, por 1 min)
-Expectativa de vida: 1.000.000 de operaciones
-Dimensiones teclado: 69*77mm
-Cable de cinta plana de 8.5 cm de largo aprox. (incluido el conector)
-Conector tipo DuPont hembra de una fila y 8 contactos con separación estándar 0.1" (2.54mm)
-Temperatura de operación: 0 a 50 °C

Figura 8. Acelerómetro MMA7361 para detectar el movimiento y/o vibración de una estructura

El MMA7361 es un acelerómetro analógico de tres ejes. Debido a que la medición de la aceleración en cada uno de los ejes se obtiene con una señal analógica, es muy fácil de usar con cualquier microcontrolador. Este acelerómetro permite medir inclinación, vibración y caída libre. Sus especificaciones son las siguientes.

-Voltaje de operación:2V~3.6V
-Consumo de corriente: 400uA
-Consumo de corriente en modo sleep: 3uA
-Alta sensibilidad: 800mV/g a 1.5G
-Rango de medida seleccionable: +-1.5G, +-6G
-Regulador de voltaje:3V/5V
-Capacidad de detección: Caída libre (salida digital)
-Filtro: Pasa bajos en las salidas analógicas
-Tiempo de encendido (rápido):5ms de tiempo de respuesta
-Autoprueba: Detección de caída libre
-Abertura del sensor: 5mm
-Salida Digital: 0V ó 5V
-Salida Analógica: 0V~5V

Y sus terminales:

*5V – Voltaje de entrada de 5 V o 3.3 V
*3V – Voltaje de entrada de 3.3 V
*Ground (GND) – Tierra común del circuito
*Eje – X – Salida analógica a lo largo del eje X
*Eje – Y – Salida analógica a lo largo del eje Y
*Eje – Z – Salida analógica a lo largo del eje Z
*Sleep (SL) – Terminal para activar el modo sleep
*Detect (0G) – Esta terminal se activa cuando se detecta caída libre.
*Sense Select (GS) – Si la terminal esta en bajo se activa el modo 1,5 g. si es alto, se cambia al modo de 6 g.
*Self Test (ST) – Este chip se ha construido en un auto-test para verificar que tanto las piezas mecánicas y eléctricas en el interior del chip están funcionando correctamente. Es útil para la calibración.
Es un acelerómetro capacitivo micromaquinado de bajo perfil y baja potencia con acondicionamiento de señal. El dispositivo consta de una celda de detección capacitiva micromaquinada de superficie (celda g) y un ASIC de acondicionamiento de señal contenido en un solo paquete. El elemento sensor se sella herméticamente al nivel de la oblea utilizando una oblea de tapa micromecanizada a granel. Puede modelarse como un conjunto de vigas unidas a una masa central móvil que se mueven entre vigas fijas. Las vigas móviles se pueden desviar de su posición de reposo sometiendo el sistema a una aceleración. A medida que se mueven las vigas unidas a la masa central, la distancia desde ellas a las vigas fijas de un lado aumentará en la misma cantidad que la distancia a las vigas fijas del otro lado disminuye. El cambio de distancia es una medida de aceleración.

Los haces de céldas g forman dos condensadores adosados. A medida que el haz central se mueve con aceleración, la distancia entre los haces cambia y el valor de cada capacitor cambiará ($C = A\varepsilon / D$). Donde A es el área del haz, ε es la constante dieléctrica y D es la distancia entre los haces. El ASIC utiliza técnicas de capacitores conmutados para medir los capacitores de celda g y extraer los datos de aceleración de la diferencia entre los dos capacitores. El ASIC también señaliza las condiciones y filtra (condensador conmutado) la señal, proporcionando un voltaje de salida de alto nivel que es ratiométrico y proporcional a la aceleración.

Tiene un modo de seleccionar la sensibilidad, seleccionado a 1.5g o 6g. Se puede cambiar usando el pin 10. También tiene un modo “desconectado” para reducir el gasto de la

batería. Se activo con el pin 7. También tiene un filtro pasa bajas incluido. Incluye radiometría, donde el voltaje de compensación de salida y la sensibilidad escalarán linealmente con el voltaje de suministro aplicado.

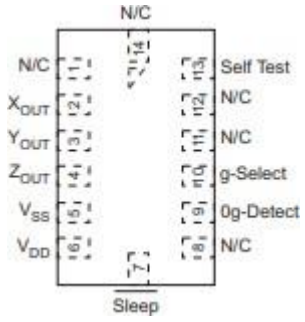

Figura 9. Pines del acelerómetro MMA7361L

# del pin	Nombre del pin	Descripción
1	N/C	Sin conexión interna Dejar desconectado
2	Xout	Voltaje de salida de dirección X
3	Yout	Voltaje de salida de dirección Y
4	Zout	Voltaje de salida de dirección Z
5	VSS	Tierra de la fuente de alimentación
6	VDD	Entrada de fuente de alimentación
7	Sleep	Pin de entrada lógica para habilitar el producto o el modo de suspensión
8	NC	Sin conexión interna Dejar desconectado
9	0g-Detect	Señal de salida lógica digital lineal de caída libre
10	g-Select	Pin de entrada lógica para seleccionar el nivel g
11	N/C	Sin conexión interna Dejar desconectado
12	N/C	Sin conexión interna Dejar desconectado
13	Selft Test	Pin de entrada para iniciar la autoprueba
14	N/C	Sin conexión interna Dejar desconectado

Tabla 1. Descripciones de los pines del acelerómetro MMA7361L

Sus características son las siguientes:
-Dimensiones: 3 x 5 x1 mm
-Bajo consumo de corriente: 400 μA
-Operación a bajo voltaje: 2.2 V – 3.6 V
-Alta sensibilidad: 800 mV/g
-Autoprueba para el diagnóstico de detección de caída libre
-Acondicionamiento de señal con filtro de paso bajo
-Bajo costo
-Diseño robusto, alta capacidad de supervivencia a impactos
-Voltaje que soporta el dieléctrico: 250 VRMS (@ 60Hz, por 1 min)
-Expectativa de vida: 1.000.000 de operaciones
-Dimensiones teclado: 69*77mm
-Cable de cinta plana de 8.5 cm de largo aprox. (incluido el conector)
-Conector tipo DuPont hembra de una fila y 8 contactos con separación estándar 0.1" (2.54mm)
-Temperatura de operación: 0 a 50 °C

● Triac Lm317

Un diac es un dispositivo semiconductor de cuatro capas y dos terminales (tiristor) que conduce corriente en una u otra dirección cuando se activa y un triac es como un diac con una terminal compuerta. Un triac puede ser disparado por un pulso de corriente en la compuerta y no requiere voltaje de ruptura para iniciar la conducción.

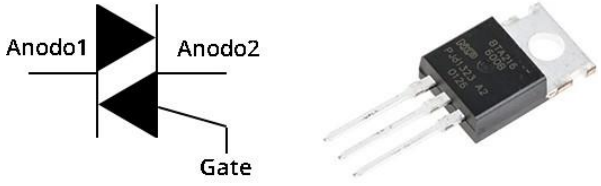

Figura 10. Triac y símbolo.

El triac puede conducir corriente en una u otra dirección cuando es activado, según la polaridad del voltaje a través de sus terminales A1, A2 y Gate. En los ánodos se coloca la corriente alterna junto con el elemento que se quiere controlar, ya sea un motor, una lampara, un horno, etc. Una vez que colocamos una corriente dentro de la terminal gate este se activa para actuar como un interruptor cerrado. El funcionamiento del triac es muy parecido al de un transistor ya que para activar estos componentes debes sobre pasar la corriente umbral en la terminal gate.

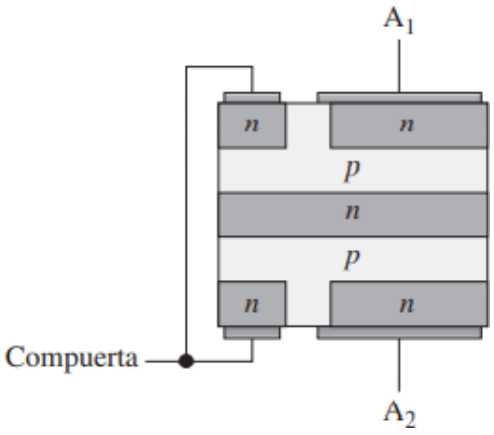

Figura 11. Construcción básica de un Triac.

El potencial de ruptura se reduce a medida que se incrementa la corriente en la compuerta. El triac deja de conducir cuando la corriente en el ánodo se reduce por debajo del valor especificado de la corriente de retención. La única forma de apagar el triac es reducir la corriente a un nivel suficientemente bajo.

● Display LCD 16x2

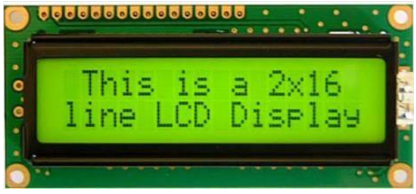

Figura 12. LCD de 16x2.

Las siglas LCD significan “Liquid Cristal Display” ó pantalla de cristal líquido. Es una pantalla plana basada en el uso de una sustancia liquida atrapada entre dos placas de vidrio, haciendo pasar por este una corriente eléctrica a una zona específica, para que así esta se vuelva opaca, y además cuenta (generalmente) con iluminación trasera.

El pin “RS” controla en que parte de la memoria LCD se están escribiendo los datos. Es aquí donde se mantiene la información que sale en la pantalla, o donde el controlador de esta busca los siguientes datos a mostrar.
El pin de “lectura/escritura” (R/W) selecciona el modo de lectura o de escritura.
El pin para habilitar “enable”, este habilita los registros.
8 pines de datos “D00-D07”, Los estados de estos pines son bits que estás escribiendo en un registro, o valores que estás leyendo.
Existe un pin “de contraste” del display.
Existe un pin “de retroiluminación” (Bklt+ y Bklt-) que le permiten controlar la retroiluminación.
Pin de alimentación (+5V y GND).

Diagrama esquemático

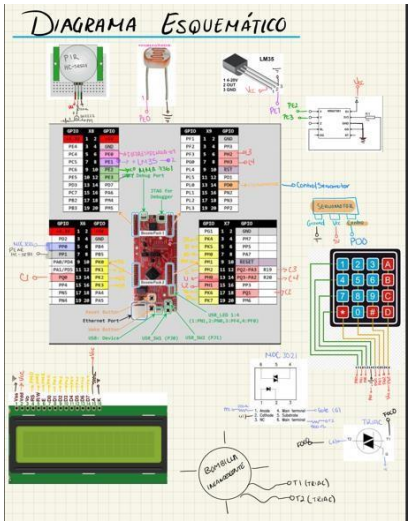

Figura 13. Diagrama esquemático del circuito resultante.

Gpios empleados

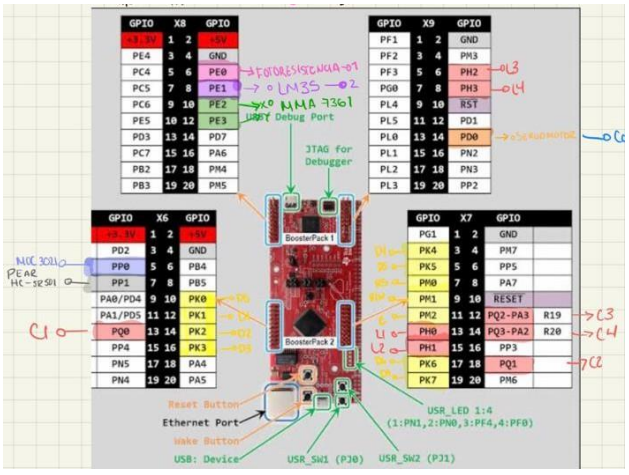

Figura 14. Gpios empleados para los componentes empleados.

Para el sensor de movimiento

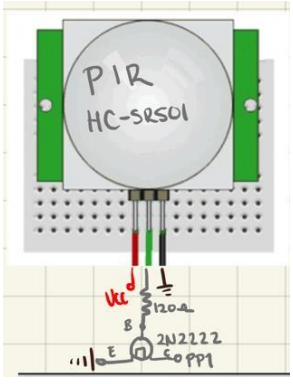

Figura15. Conexión del sensor de movimiento PIR HC-5R501.

Para la fotoresistencia

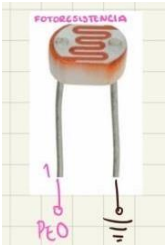

Figura 16. Conexión de la fotorresistencia.

Para el Lm35

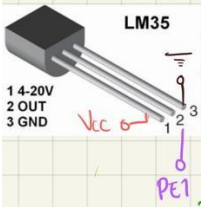

Figura 17. Sensor de temperatura.

Para el MMA7361

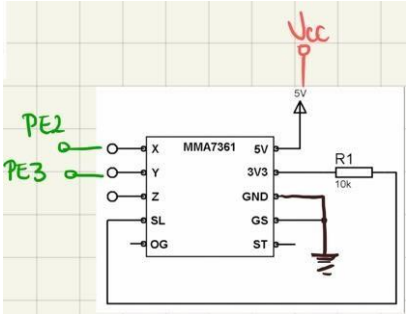

Figura 18. Acelerometro MMA-7361

Para el servomotor

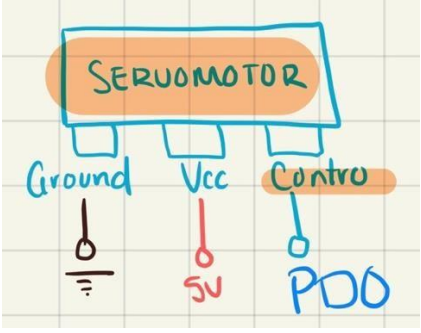

Figura 19. Servomotor para abrir la cajuela.

Para el teclado matricial 4x4

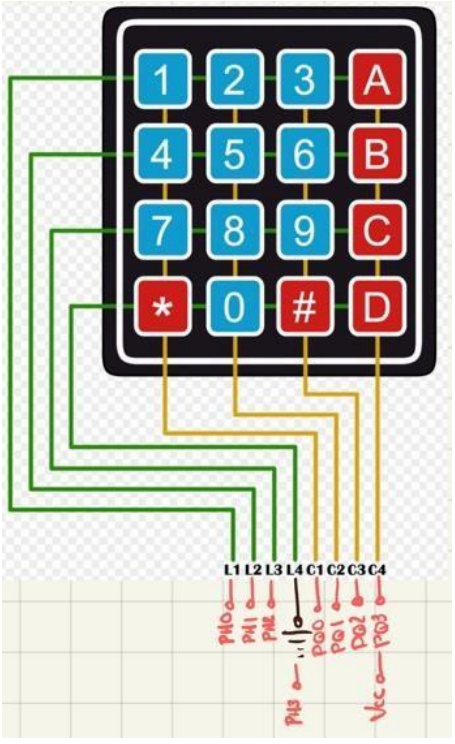

Figura 20. Teclado matricial 4x4 y sus conexiones.

Para el Moc 3021

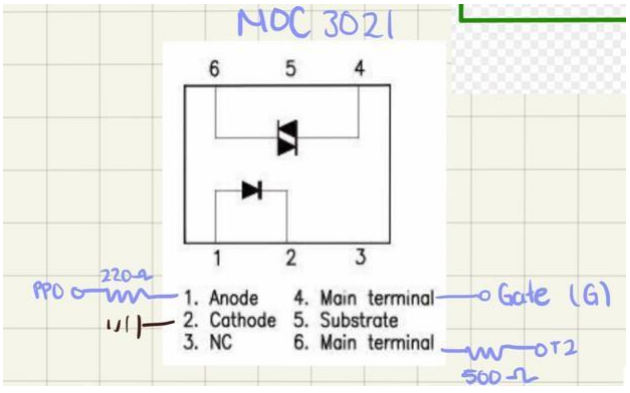

Figura 21. Conexiones del MOC 3021

Para el display LCD de 16x2

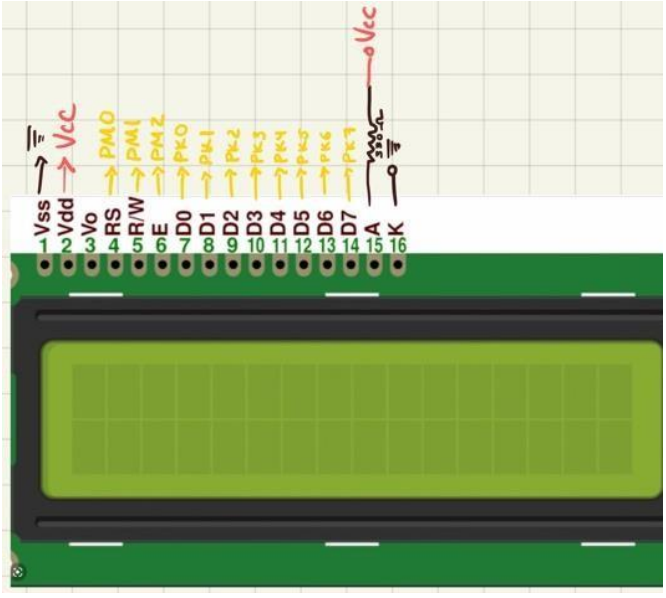

Figura 22. Display LCD y sus conexiones d

Costo del dispositivo

Componente	Precio $
Tiva C Series TM4C1294NCPDT	500
LM35	39
Fotoresistencia	42
Moc 3021	150
Servomotor	50
HC-SR501	30
Teclado matricial 4x4	48
Acelerómetro MMA7361	45
Triac lm317	28
Display LCD 16x2	52
Total =	$984.00

Tabla 2. Costo del sistema

El valor de los componentes se queda en $984, se despreciaron los precios de componentes extra y de mano de obra del ingeniero debido a que su uso en este caso es para uno de los integrantes.

En caso de ser para otra persona se agregan $60 se componentes extra y $1000 libres para la persona encargada de diseñar e implementar el sistema de seguridad.

Código

```
#include   <stdint.h>
#include <stdbool.h>
#include  "inc/tm4c1294ncpdt.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"



#define ValPLLFREQ0 0X00800060;
#define ValPLLFREQ1 0X00000004;

   uint32_t ValorADC;
   uint32_t ui32Loop;
   uint32_t lm35=0;
   uint32_t Fotoresistencia=0;
   uint32_t AcelerometroX=0;
   uint32_t AcelerometroY=0;
   int promediolm35=0;
   int promedioFotoresistencia=0;
   int promedioAcelerometroX=0;
   int promedioAcelerometroY=0;
   float ValorADCf;
   char Codigo="a";
   int c=0;
   int r=0;
#define brs 0b00000001
#define brw 0b00000010
#define benable 0x04 //0b00000100
#define vartmpDecenas 0x00//esta variable inicializa el tiempo en 0
#define vartmpUniades 0x00//esta variable inicializa el tiempo en 0
//
//FUNCIONES//
//configuracionPuertos habilita los poertos, los dispone como salidas digitales
//inicializamos rs y rw en 0, asi como todos los puertos k
 void configuracionPuertos()
 {
    /*; habilitar el PORTK y PORTM. bits 11 y 9 del RCGCGPIO 0000 1010 0000 0000

      inipro nop

       ;rgcgcpio 0x400f.e608
         mov r0,#0xe608
         movt r0,#0x400f
            mov  r1,#0xa00  ;0000 1010 0000 0000
            str  r1,[r0]

      ;prgpio Base 0x400F.E000
      ;Offset 0xA08

           mov r0,#0xea08 ;espera a que los puertos k y m estén habilitados
```

```
        movt r0,#0x400f
    eapkm  ldr  r1,[r0]
        ands r1,#0xa00
        ;cmp r1,#0xa00
        ;bne eapkm
        beq  eapkm*/
    //#define          SYSCTL_RCGCGPIO_R9          Y
SYSCTL_RCGCGPIO_R11,          K          Y          M

RESPECTIVAMENTE 0B0000 1010 0000 0000
    //TENEMOS QUE ACTIVAR LOS DOS PUERTOS Y
ESPERAR A QUE UNO ESTE ABIERTO
    //el SYSCTL_RCGCGPIO_R va a guardar la
informacion de los puertos a encender
    //si le das el valos directo en hexadecimal a 8 bits exacto
de los puertos que quieres tambien
    //los activa
    //primero enmascaro con or el valor que quiero que
tenga, asi no afecto el valor de los demas registros
    SYSCTL_RCGCGPIO_R                              |=
SYSCTL_RCGCGPIO_R9 ;
    //comparo con un and
    //todos los bitsque tengan cero obtendran cero de
resultado
    //todos los bits que estenen 1, si la otra variable es 1,
deja el 1
    //asi el resultado ya no es 0
    //aun asi esta operacion se realiza bit a bit por lo que
cuando tetecte el primer puerto encencido
    //ya no verificara el segundo
    while((SYSCTL_PRGPIO_R                         &
SYSCTL_RCGCGPIO_R9) == 0);
    SYSCTL_RCGCGPIO_R                              |=
SYSCTL_RCGCGPIO_R11 ;
    while((SYSCTL_PRGPIO_R                         &
SYSCTL_RCGCGPIO_R11) == 0);
    SYSCTL_RCGCGPIO_R                              |=
SYSCTL_RCGCGPIO_R1 ;
    while((SYSCTL_PRGPIO_R                         &
SYSCTL_RCGCGPIO_R1) == 0);

    /*
        ;direccion base de K 0X4006.1000

        ;configurar PORTK como salida en todos los bits
        ;offset de dir 0x400

            mov r0,#0x1400
            movt r0,#0x4006
            mov  r1,#0xff ;1111 1111
            str  r1,[r0]

        ;configurar PORTK como digital en todos los bits
        ;offset de den 0x51c  DEN


            mov r0,#0x151c
            movt r0,#0x4006
            mov r1,#0xff ;
            str  r1,[r0]

        ;direccion  base PORTK 0x4006.1000
        ;ENCENDER (poner a 1)los PRIMEROS DOS bits
de N
        ;PORTK DATA offset 0x3fc

            mov  r0,#0x13fc
            movt r0,#0x4006
            mov  r1,#0x00 ;
            str  r1,[r0] */

    // pongo el puerto k como salida y digital
    //todos sus puertos
    GPIO_PORTK_DIR_R = 0B11111111;
    GPIO_PORTK_DEN_R = 0B11111111;
    GPIO_PORTK_DATA_R = 0X00;
    //Pongo el puerto m como salida y digital
    // solo los puertos 0 y 2
    GPIO_PORTM_DIR_R = 0B00000101;
    GPIO_PORTM_DEN_R = 0B00000101;
    GPIO_PORTM_DATA_R = 0X05;
    //CONFIGURACION PUERTOS B
    GPIO_PORTB_AHB_DIR_R = 0B00000000;
    GPIO_PORTB_AHB_DEN_R = 0B00000011;
    GPIO_PORTB_AHB_DATA_R = 0X00;
 }
 void Cronometro(float seg)
 {
     //Escribimos en el puerto 0000 0000
     GPIO_PORTK_DATA_R |=0X00;
     float VelCPU = 62.5*(10E-9);//A 16 Mhz T=62.5 nS
     int TICKS = seg/VelCPU;//obtenemos el numero para
el systick
     NVIC_ST_RELOAD_R=TICKS;
     //Ponemos A NUESTRA DISPOSICION CLOCK
SOURC, SIN INTERRUPCIONES(INTEN), Y
ACTIVADO (ENABLE)
     NVIC_ST_CTRL_R=0X05; //CLK_SRC  INTEN EN
0101
    // checamos la bandera de count posicion 16
     while (( NVIC_ST_CTRL_R&0x10000)==0); // 0001
0000 0000 0000 0000
   //Cuando pase este ciclo el tiempo acabara
 }

 void clearPorts()
 {
   /*;esta rutina pone a cero todos los
   ;bits de los puertos utilizados
   ;direccion base PORTM 0x4006.3000
   ;PORTM DATA offset 0x3fc
   clrports mov  r0,#0x33fc  ;cero a c
        movt r0,#0x4006
        mov r1,#0x00 ;
        str  r1,[r0]

   ;direccion base PORTk 0x4006.1000
   ;PORTM DATA offset 0x3fc
        mov r0,#0x13fc ;cero a o
        movt r0,#0x4006
        mov r1,#0x00 ;
        str  r1,[r0]
        bx LR*/
```

```
    GPIO_PORTK_DATA_R = 0X00;
    GPIO_PORTM_DATA_R = 0X00;
 }
//esta funcion manda el comando al display el dato ya tiene
que estar registrado en el registro destino
void EscribirComando()
{
   GPIO_PORTM_DATA_R |= 0x04;
   Cronometro(0.02);
   GPIO_PORTM_DATA_R &= 0xfb;
}
void inicioDisplay()
{
   /*;Esta rutino asegura el inicio seguro del display segun
el manual
;eL PRIMER PASO ES  PRENDER EL DISPLAY
iniciodisplay mov r6,#0x0f
        push {lr}
        bl escom
        pop {lr}
;tenemos que esperar mas de 15 ml, lo calcule a 20ms
        push {lr}
        bl retdisp
        pop {lr}
;tenemos que poner en funcion set
;enviaremos 0011 1000 = 38
        mov  r6,#0x38
        push {lr}
        bl  escom
        pop {lr}
;esperamos 4.1 microsegundos, que son =0.0041ms
;usaremos 20ms
        push  {lr}
        bl retdisp
        pop {lr}
;tenemos que poner en funcion set
;enviaremos 0011 1000 = 38
        mov  r6,#0x38
        push {lr}
        bl  escom
        pop {lr}
;esperamos 100 microsegundos, que son =0.1ms
;usaremos 20ms
        push  {lr}
        bl retdisp
        pop {lr}
;tenemos que poner en funcion set
;enviaremos 0011 1000 = 38
        mov  r6,#0x38
        push {lr}
        bl  escom
        pop {lr}

;tenemos que poner en funcion set
;enviaremos 0011 1000 = 38
        mov  r6,#0x38
        push {lr}
        bl  escom
        pop {lr}
;tenemos que mandar Display ON
;0000 1110 = 0c
        mov  r6,#0x0f
        push {lr}
        bl  escom
        pop {lr}
;tenemos que mandar Display Clear
;0000 0001 = 01
        mov r6,#0x01
        push {lr}
        bl escom
        pop {lr}
;tenemos que mandar Entry Mode Set
;0000 0110 = 06
        mov r6,#0x06
        push {lr}
        bl escom
        pop {lr}
        bx  lr*/

  //eL PRIMER PASO ES  PRENDER EL DISPLAY
  GPIO_PORTK_DATA_R = 0X0f;
  EscribirComando();
  //tenemos que esperar mas de 15 ml, lo calcule a 20ms
  Cronometro(0.02);
  EscribirComando();
  //tenemos que poner en funcion set
  //enviaremos 0011 1000 = 38
  GPIO_PORTK_DATA_R = 0X38;
  EscribirComando();
  //;esperamos 4.1 microsegundos, que son =0.0041ms
  //;usaremos 20ms
  Cronometro(0.02);
  EscribirComando();
  //;tenemos que poner en funcion set
  //;enviaremos 0011 1000 = 38
  GPIO_PORTK_DATA_R = 0X38;
  EscribirComando();
  //;esperamos 100 microsegundos, que son =0.1ms
  //;usaremos 20ms
  Cronometro(0.02);
  EscribirComando();
  //;tenemos que poner en funcion set
  //;enviaremos 0011 1000 = 38
  GPIO_PORTK_DATA_R = 0X38;
  EscribirComando();
  //;tenemos que poner en funcion set
  //;enviaremos 0011 1000 = 38
  GPIO_PORTK_DATA_R = 0X38;
  EscribirComando();
  //;tenemos que mandar Display ON
  //;0000 1110 = 0c
  GPIO_PORTK_DATA_R = 0X0c;
  EscribirComando();
  //;tenemos que mandar Display Clear
  //;0000 0001 = 01
  GPIO_PORTK_DATA_R = 0X01;
  EscribirComando();
  //;tenemos que mandar Entry Mode Set
  //;0000 0110 = 06
  GPIO_PORTK_DATA_R = 0X06;
  EscribirComando();
}
void EscribirDatos()
{
```

```
    GPIO_PORTM_DATA_R |= brs;
    GPIO_PORTM_DATA_R |= benable;
    Cronometro(0.02);
    GPIO_PORTM_DATA_R &= 0xfa;
}

void Abecedario(char letra)
{
    int e=0;
    switch( letra )
    {
        case 'A':
            GPIO_PORTK_DATA_R = 0x41;
            EscribirDatos();
            break;
        case 'a':
            GPIO_PORTK_DATA_R = 0x61;
            EscribirDatos();
            break;
        case 'B':
            GPIO_PORTK_DATA_R = 0x42;
            EscribirDatos();
            break;
        case 'b':
            GPIO_PORTK_DATA_R = 0x62;
            EscribirDatos();
            break;
        case 'C':
            GPIO_PORTK_DATA_R = 0x43;
            EscribirDatos();
            break;
        case 'c':
            GPIO_PORTK_DATA_R = 0x63;
            EscribirDatos();
            break;
        case 'D':
            GPIO_PORTK_DATA_R = 0x44;
            EscribirDatos();
            break;
        case 'd':
            GPIO_PORTK_DATA_R = 0x64;
            EscribirDatos();
            break;
        case 'E':
            GPIO_PORTK_DATA_R = 0x45;
            EscribirDatos();
            break;
        case 'e':
            GPIO_PORTK_DATA_R = 0x65;
            EscribirDatos();
            break;
        case 'F':
            GPIO_PORTK_DATA_R = 0x46;
            EscribirDatos();
            break;
        case 'f':
            GPIO_PORTK_DATA_R = 0x66;
            EscribirDatos();
            break;
        case 'G':
            GPIO_PORTK_DATA_R = 0x47;
            EscribirDatos();
            break;
        case 'g':
            GPIO_PORTK_DATA_R = 0x67;
            EscribirDatos();
            break;
        case 'H':
            GPIO_PORTK_DATA_R = 0x48;
            EscribirDatos();
            break;
        case 'h':
            GPIO_PORTK_DATA_R = 0x68;
            EscribirDatos();
            break;
        case 'I':
            GPIO_PORTK_DATA_R = 0x49;
            EscribirDatos();
            break;
        case 'i':
            GPIO_PORTK_DATA_R = 0x69;
            EscribirDatos();
            break;
        case 'J':
            GPIO_PORTK_DATA_R = 0x4a;
            EscribirDatos();
            break;
        case 'j':
            GPIO_PORTK_DATA_R = 0x6a;
            EscribirDatos();
            break;
        case 'K':
            GPIO_PORTK_DATA_R = 0x4b;
            EscribirDatos();
            break;
        case 'k':
            GPIO_PORTK_DATA_R = 0x6b;
            EscribirDatos();
            break;
        case 'L':
            GPIO_PORTK_DATA_R = 0x4c;
            EscribirDatos();
            break;
        case 'l':
            GPIO_PORTK_DATA_R = 0x6c;
            EscribirDatos();
            break;
        case 'M':
            GPIO_PORTK_DATA_R = 0x4d;
            EscribirDatos();
            break;
        case 'm':
            GPIO_PORTK_DATA_R = 0x6d;
            EscribirDatos();
            break;
        case 'N':
            GPIO_PORTK_DATA_R = 0x4e;
            EscribirDatos();
            break;
        case 'n':
            GPIO_PORTK_DATA_R = 0x6e;
```

```
      EscribirDatos();
      break;
   case 'O':
      GPIO_PORTK_DATA_R = 0x4f;
      EscribirDatos();
      break;
   case 'o':
      GPIO_PORTK_DATA_R = 0x6f;
      EscribirDatos();
      break;
   case 'P':
      GPIO_PORTK_DATA_R = 0x50;
      EscribirDatos();
      break;
   case 'p':
      GPIO_PORTK_DATA_R = 0x70;
      EscribirDatos();
      break;
   case 'Q':
      GPIO_PORTK_DATA_R = 0x51;
      EscribirDatos();
      break;
   case 'q':
      GPIO_PORTK_DATA_R = 0x71;
      EscribirDatos();
      break;
   case 'R':
      GPIO_PORTK_DATA_R = 0x52;
      EscribirDatos();
      break;
   case 'r':
      GPIO_PORTK_DATA_R = 0x72;
      EscribirDatos();
      break;
   case 'S':
      GPIO_PORTK_DATA_R = 0x53;
      EscribirDatos();
      break;
   case 's':
      GPIO_PORTK_DATA_R = 0x73;
      EscribirDatos();
      break;
   case 'T':
      GPIO_PORTK_DATA_R = 0x54;
      EscribirDatos();
      break;
   case 't':
      GPIO_PORTK_DATA_R = 0x74;
      EscribirDatos();
      break;
   case 'U':
      GPIO_PORTK_DATA_R = 0x55;
      EscribirDatos();
      break;
   case 'u':
      GPIO_PORTK_DATA_R = 0x75;
      EscribirDatos();
      break;
   case 'V':
      GPIO_PORTK_DATA_R = 0x56;
      EscribirDatos();
      break;
   case 'v':
      GPIO_PORTK_DATA_R = 0x76;
      EscribirDatos();
      break;
   case 'W':
      GPIO_PORTK_DATA_R = 0x57;
      EscribirDatos();
      break;
   case 'w':
      GPIO_PORTK_DATA_R = 0x77;
      EscribirDatos();
      break;
   case 'X':
      GPIO_PORTK_DATA_R = 0x58;
      EscribirDatos();
      break;
   case 'x':
      GPIO_PORTK_DATA_R = 0x78;
      EscribirDatos();
      break;
   case 'Y':
      GPIO_PORTK_DATA_R = 0x59;
      EscribirDatos();
      break;
   case 'y':
      GPIO_PORTK_DATA_R = 0x79;
      EscribirDatos();
      break;
   case 'Z':
      GPIO_PORTK_DATA_R = 0x5a;
      EscribirDatos();
      break;
   case 'z':
      GPIO_PORTK_DATA_R = 0x7a;
      EscribirDatos();
      break;
   case ' ':
      GPIO_PORTK_DATA_R = 0x20;
      EscribirDatos();
      break;
   default :
      e++;
  }

}
void EscribirDisplay(char texto[])
{
   //GPIO_PORTK_DATA_R = 0X84;
   //EscribirComando();
   //GPIO_PORTK_DATA_R = 0xc7;
   char DatoArreglo = texto[0];
   int i=0;

   GPIO_PORTK_DATA_R = 0X80;
   EscribirComando();
   while(DatoArreglo!='\0')
    {

      //UART0_DR_R=DatoArreglo;
      Abecedario(DatoArreglo);
      i++;
      DatoArreglo = texto[i];
```

```
      if(i==16)
      {
         GPIO_PORTK_DATA_R = 0Xc0;
         EscribirComando();
      }
     }

}


void inicioLucesYMov()
{
   //habilitamos el puerto p0
   SYSCTL_RCGCGPIO_R                                    |=
SYSCTL_RCGCGPIO_R13;
   ValorADC=65565;
   GPIO_PORTP_DATA_R = 0X00;
   GPIO_PORTP_DEN_R = 0X03; //HABILITO EL PIN
DIGITAL p0 y p1
   GPIO_PORTP_DIR_R = 0X01; //lo uso como salida p0 y
como entrada p1
   GPIO_PORTP_DATA_R = 0X00;
}
void inicioServo()
{
   SYSCTL_RCGCGPIO_R                                    |=
SYSCTL_RCGCGPIO_R3;//habilito el puerto D
   SYSCTL_RCGCTIMER_R |= 0X01;//habilito el timer 0
correspondiente a d0 t0ccp0

   GPIO_PORTD_AHB_DEN_R |= 0x01; //BIT 1
DIGITAL
   GPIO_PORTD_AHB_DIR_R |= 0x01; //bit 1 SALIDA
   GPIO_PORTD_AHB_DATA_R = 0x00; // SALIDA A 0
   GPIO_PORTD_AHB_AFSEL_R = 0x01; //FUNCION
ALTERNA EN BIT 1 0000 0001
   GPIO_PORTD_AHB_PCTL_R = 0x00000003;
//DIRIGIDO A T0CCP0

   TIMER0_CTL_R=0X00000000; //DESHABILITA
TIMER EN LA CONFIGURACION
   TIMER0_CFG_R= 0X00000004; //CONFIGURAR
PARA 16 BITS
   TIMER0_TAMR_R= 0X0000000A; //CONFIGURAR
PARA MODO PWM, MODO PERIODICO CUENTA
HACIA ABAJO
   TIMER0_TBMR_R= 0X0000000A; //CONFIGURAR
PARA MODO PWM, MODO PERIODICO CUENTA
HACIA ABAJO
   //Para configurara el pmw se tomaron en cuenta las
siguients consideraciones
   //1/20ms=50 hz, esa es la frecuencia que nececitamos
   // 16000000/50 = 320000 = 4e200
   //Con ese valor Ya tengo la señal de 20 ms que me pide el
fabricante pero como hago para girarlo
   //me dicen que para que gire tengo que mandarle un pulso
con cierta duracion
   //con 1ms gira completamente a la izquierda
   //con 1.5ms gira al centro
   //con 2ms se va completamente a la derecha
   //Sacando los porcentajes de la señal de 20 ms
   //1ms = 5% de 20 ms
   //1.5ms = 7.5% de 20 ms
   //2 ms = 10% de 20 ms
   //Para calcular el valor del match tenemos que sacarlo de
su valor
   //320*0.1=32000 = 10%
   //320*0.075 =24000 = 7.5%
   //320*0.05 =16000 = 5%
   // En hexacecimal
   //16000 = 3e80
   //24000 = 5DC0
   // 32000 = 7D00
   TIMER0_TAILR_R= 0xe200; //
   TIMER0_TAMATCHR_R =0x3e80; // 100 %
   TIMER0_TAPR_R= 0X04; // preescalador
   TIMER0_CTL_R |= 0X00000041; //HABILITA TIMER
A
}
void inicioDispositivosAnalogicos()
{
   //Habilito los puertos E
   //voy a usar desde E0-E5
   SYSCTL_RCGCGPIO_R                                    |=
SYSCTL_RCGCGPIO_R4;
   //reloj para el adc
   SYSCTL_RCGCADC_R |= SYSCTL_RCGCADC_R0;
   ValorADC=65565; //tiempo para que el reloj llegue a los
modulos
   //EL PLL es necesario activarlo, asi lo pide
TexasInstruments
   SYSCTL_PLLFREQ1_R = ValPLLFREQ1;
   SYSCTL_PLLFREQ0_R = ValPLLFREQ0;
   while((SYSCTL_PLLSTAT_R&0x01)==0);
   //Esto no se porque se repite
   SYSCTL_RCGCGPIO_R                                    |=
SYSCTL_RCGCGPIO_R4;
   SYSCTL_RCGCADC_R |= SYSCTL_RCGCADC_R0;
   ValorADC=65565; //tiempo para que el reloj llegue a los
modulos

   //inicializar PE.0 para que sea AIN3
   //inicializar PE.1 para que sea AIN2
   //inicializar PE.2 para que sea AIN1
   //inicializar PE.3 para que sea AIN0
   //inicializar PE.4 para que sea AIN9
   //inicializar PE.5 para que sea AIN8

   GPIO_PORTE_AHB_AMSEL_R |=0X3f; //habilita
modulo analogico de E0-E5 0011 1111
   GPIO_PORTE_AHB_DEN_R &= ~0X3f; //desahabilita
buffer digital

   // inicializar adc
   ADC0_SSPRI_R=0X00003210; //se queda con las
mismas prioridades p1099
   //ADC0_CC_R=0X170; //RELOJ DEL ADC EL VCO del
PLL /24
   ADC0_PC_R=0x07;//1Ms/S velocidad de la muestra
p1159
   //ADC0_ACTSS_R &= ~8; //deshabilitar SS3 (o todos
los secuenciadores)
```

```
   ADC0_ACTSS_R   = 0;         //durante la configuración p1077
   //ADC0_EMUX_R     &= ~0XF000;     //conversión por software para todos los secuenciadores
   ADC0_EMUX_R = 0X0000;  //inicio  de la conversión por software con bit SSn en el ADCPSSI, p1091
   //aqui establece el secuenciador coneccion con el pin AIN, para hacer sus conversiones,
   //si eemux3 es 0, el valor del bit corresponde al valor de ainx,[0-15], For example, if the
   //MUX3 field is 0x0, AIN0 is selected.
   //si eemux3 es 1, el valor del bit corresponde al valor de ainx [16-19]
   //For example, if the
   //MUX3 field is 0x0, AIN16 is selected.
   ADC0_SSEMUX0_R   = 0;        //una vez establecido, el conteo, hacia arriba
   ADC0_SSMUX0_R  = 0;         //ain0 seleccionado
   ADC0_SSEMUX1_R  = 0;         //una vez establecido, el conteo, hacia arriba
   ADC0_SSMUX1_R  = 1;         //ain1 seleccionado
   ADC0_SSEMUX2_R  = 0;         //una vez establecido, el conteo, hacia arriba
   ADC0_SSMUX2_R  = 2;         //ain2 seleccionado
   ADC0_SSEMUX3_R  = 0;         //una vez establecido, el conteo, hacia arriba
   ADC0_SSMUX3_R  = 3;         //ain3 seleccionado
   ADC0_SSCTL0_R    |= 2;          ////poner bandera para terminar a la 1a muestra p1142  0010
   ADC0_SSCTL1_R    |= 2;          ////poner bandera para terminar a la 1a muestra p1142 0010
   ADC0_SSCTL2_R    |= 2;          ////poner bandera para terminar a la 1a muestra p1142 0010
   ADC0_SSCTL3_R    |= 2;          ////poner bandera para terminar a la 1a muestra p1142 0010
   //ADC0_SSCTL2_R    |= 0x0a;          //tomar lectura del sensor de temperatura
   ADC0_ACTSS_R   |= 0X0f;        //habilitar secuenciador 0 del ADC0 p1077
   SYSCTL_PLLFREQ0_R=0; //DESHABILITA EL PLL
   SYSCTL_PLLFREQ1_R=0;

}
void LeerPuertosAnalogicos()
{
   /*uint32_t  lm35=0;
   uint32_t Fotoresistencia=0;
   uint32_t AcelerometroX=0;
   uint32_t AcelerometroY=0;
   */
   ADC0_PSSI_R    |= 0x0f;                  //EMPEZAR SECUENCIA DE CONVERSION de todos los secuenciadores p1103
   while((ADC0_RIS_R  &  0x0f  ==  0));    //espera por conversión completa  p1079
   Fotoresistencia=ADC0_SSFIFO0_R;
   lm35= ADC0_SSFIFO1_R;
   AcelerometroX=ADC0_SSFIFO2_R;
   AcelerometroY=ADC0_SSFIFO3_R;
   ADC0_ISC_R = 0x0f;
}
void prenderLuces()
{
   GPIO_PORTP_DATA_R = 0X01;
}
void apagarLuces()
{
   GPIO_PORTP_DATA_R = 0X00;
}

void vigilarMovimiento()
{


      //EscribirDisplay("    Se Roban      la moto");
      if(GPIO_PORTP_DATA_R == 0x02){
      prenderLuces();
      SysCtlDelay(4800);
      apagarLuces();
      SysCtlDelay(4800);}



}
void girarServoIzquierda()
{
   //TIMER0_CTL_R    |=    0X00000041;    //HABILITA TIMER A
   SysCtlDelay(10000);
   TIMER0_TAMATCHR_R=0x3e80;
   SysCtlDelay(10000);
   //TIMER0_CTL_R=0X00000000;       //DESHABILITA TIMER EN LA CONFIGURACION
}
void girarServoDerecha()
{
   //TIMER0_CTL_R    |=    0X00000041;    //HABILITA TIMER A
   SysCtlDelay(10000);
   TIMER0_TAMATCHR_R=0x7D00;
   SysCtlDelay(10000);
   //TIMER0_CTL_R=0X00000000;       //DESHABILITA TIMER EN LA CONFIGURACION
}



void AbrirCajuela()
{
   EscribirDisplay(" Codigo");
   CaracterisarTeclado();

}
void PromediarValoresAnalogicos()
{
  /* int promediolm35=0;
   int promedioFotoresistencia=0;
   int promedioAcelerometroX=0;
   int promedioAcelerometroY=0;
   uint32_t lm35=0;
   uint32_t Fotoresistencia=0;
   uint32_t AcelerometroX=0;
   uint32_t AcelerometroY=0;*/
   int i=0;
   for(i=0;i<10;i++)
   {
```

```
      LeerPuertosAnalogicos();
      promedioAcelerometroX+=AcelerometroX;
      promedioAcelerometroY+=AcelerometroY;
   }
   promedioAcelerometroY=promedioAcelerometroY/10;
   promedioAcelerometroX= promedioAcelerometroX/10;

}
void alarma()
{

   EscribirDisplay("Protegido");

   vigilarMovimiento();
   //PromediarValoresAnalogicos();
   LeerPuertosAnalogicos();
   /*if(AcelerometroY>2000 | AcelerometroY<1300)
   {
      EscribirDisplay("    Se Roban       la moto");
      while(GPIO_PORTB_AHB_DATA_R  != 0x03){
      prenderLuces();
      SysCtlDelay(4800);
      apagarLuces();
      SysCtlDelay(4800);
      }
   }*/
   if(AcelerometroX>400 | AcelerometroX<200){

     //EscribirDisplay("    Se Roban       la moto");

         prenderLuces();
         SysCtlDelay(4800);
         apagarLuces();
         SysCtlDelay(4800);
   }
}
void tiempo(){
   int i=0;
   while(i<1500000){i++;}



}
void LeerPosicionColumna(int Columna)
{
   //while(1){
   int Dato="";
   Dato = GPIO_PORTH_AHB_DATA_R;
   switch(Columna) {
   case 0:
      if(Dato == 1){//ES 1
         Codigo='1';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 2){//ES 4
         Codigo='4';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 4){//ES 7
         Codigo='7';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 8){//ES *
         Codigo='*';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      break;
   case 1:
         if(Dato == 1){//ES 1
            Codigo='2';
            VerificarCodigo(Codigo);
            c++;
            tiempo();
         }
         if(Dato == 2){//ES 4
            Codigo='5';
            VerificarCodigo(Codigo);
            c++;
            tiempo();
         }
         if(Dato == 4){//ES 7
            Codigo='8';
            VerificarCodigo(Codigo);
            c++;
            tiempo();
         }
         if(Dato == 8){//ES *
            Codigo='0';
            VerificarCodigo(Codigo);
            c++;
            tiempo();
         }
         break;
   case 2:
      if(Dato == 1){//ES 1
         Codigo='3';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 2){//ES 4
         Codigo='6';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 4){//ES 7
         Codigo='9';
         VerificarCodigo(Codigo);
         c++;
         tiempo();
      }
      if(Dato == 8){//ES *
         Codigo='#';
```

```
        VerificarCodigo(Codigo);
        c++;
        tiempo();
      }
      break;
   case 3:
      if(Dato == 1){//ES 1
        Codigo='A';
        VerificarCodigo(Codigo);
        c++;
        tiempo();
      }
      if(Dato == 2){//ES 4
        Codigo='B';
        VerificarCodigo(Codigo);
        c++;
        tiempo();
      }
      if(Dato == 4){//ES 7
        Codigo='C';
        VerificarCodigo(Codigo);
        c++;
        tiempo();
      }
      if(Dato == 8){//ES *
        Codigo='D';
        VerificarCodigo(Codigo);
        c++;
        tiempo();
      }
      break;
    default: Columna=0;
   }

   //}
}

void VerificarCodigo(char dato)
{
char Codigo1[]="5712";
//int i=0;

   if(dato== Codigo1[c]){
   r++;}
  // i++;



 if(r==4)
 {
    GPIO_PORTK_DATA_R = 0X01;
    EscribirComando();
    EscribirDisplay("Abierta");
    girarServoIzquierda();

    SysCtlDelay(10000);
    girarServoDerecha();
    c=-1;
    r=0;
 }
 if(c==4)
  {
     GPIO_PORTK_DATA_R = 0X01;
     EscribirComando();
     EscribirDisplay("Mal");
     SysCtlDelay(10000);
     girarServoDerecha();
     girarServoIzquierda();
     c=-1;
     r=0;
  }
}
void TecladoIni()
{
   SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R7
|SYSCTL_RCGCGPIO_R14;
   //Configuracion puerto H
   GPIO_PORTH_AHB_DATA_R = 0X00; //PREPARA
PUERTO n LEDS 0 Y 1
   GPIO_PORTH_AHB_DEN_R = 0X0f; //Habilita los bits
0000 1111
   GPIO_PORTH_AHB_DIR_R = 0X00; //En 0 los bits son
de entrada (input)

   //Configuracion puerto Q
   GPIO_PORTQ_DATA_R = 0X00; //PREPARA
PUERTO n LEDS 0 Y 1
   GPIO_PORTQ_DEN_R = 0X0f; //Habilita los bits 0000
1111
   GPIO_PORTQ_DIR_R = 0X0f; //En 1 los bits son de
salida
}
void CaracterisarTeclado()
{
   //while(1)
      //{
        GPIO_PORTQ_DATA_R = 0X01;
        LeerPosicionColumna(0);
        GPIO_PORTQ_DATA_R = 0X00;
        GPIO_PORTQ_DATA_R = 0X02;

        LeerPosicionColumna(1);
        GPIO_PORTQ_DATA_R = 0X00;
        GPIO_PORTQ_DATA_R = 0X04;
        LeerPosicionColumna(2);
        GPIO_PORTQ_DATA_R = 0X00;
        GPIO_PORTQ_DATA_R = 0X08;
        LeerPosicionColumna(3);
        GPIO_PORTQ_DATA_R = 0X00;
      //}
}

int main(void)
{
   inicioDispositivosAnalogicos();
   inicioServo();
   inicioLucesYMov();
   configuracionPuertos();
   clearPorts();
   inicioDisplay();
   TecladoIni();

   //LeerPuertosAnalogicos()
   //prenderLuces()
   //apagarLuces()
   //vigilarMovimiento()
   //girarServoIzquierda()
   //girarServoDerecha()
```

```
   while(1)
   {
   if(GPIO_PORTB_AHB_DATA_R  == 0x01)
   {
      alarma();
   }
   if(GPIO_PORTB_AHB_DATA_R  == 0x00)
   {
      AbrirCajuela();
   }
   }
}
```

REFERENCIAS

[1] Floyd, T. L. (2007). *Electronic Devices: Conventional Current Version* (8 Har/Cdr ed., Vol. 1). Recuperado de https://www.academia.edu/43939690/Dispositivos_Electrónicos_Dispositivos_Electrónicos

[2] z Acosta, A., & WooCommerce, S. C. C. Y. (2019, 5 junio). Servomotores – T-Bem. Recuperado 8 de febrero de 2021, de https://teslabem.com/blog/servomotores/

[3] SG90 Servo Datasheet pdf - Micro Servo. Equivalent, Catalog. (2014, 1 enero). Recuperado 8 de febrero de 2021, de https://datasheetspdf.com/pdf/791970/TowerPro/SG90/1

[4] PXN. (2008–2021). *Three Axis Low-g Micromachined Accelerometer* [Freescale Semiconductor Technical Data]. Recuperado de https://www.nxp.com/docs/en/data-sheet/MMA7361L.pdf

[5] Latam, M. (2020, 14 marzo). LDR. Recuperado de https://www.mecatronicalatam.com/es/tutoriales/sensores/sensor-de-luz/ldr/

[6] CarrodElectronica. (2014, 1 enero). Optoacoplador MOC3021M Salida Triac. Recuperado 9 de febrero de 2021, de https://www.carrod.mx/products/optoacoplador-moc3021-salida-triac#:%7E:text=Optoacoplador%20MOC3021M%20Salida%20Triac,-OPTMOC3021&text=Se%20suelen%20utilizar%20para%20aislar,un%20interruptor%20bilateral%20de%20silicio.

[7] alldatasheet.com. (2012–2021). *LM35 pdf, LM35description,LM35 datasheets, LM35 view ::: ALLDATASHEET :::* [LM35 Datasheet]. Recuperado de https://pdf1.alldatasheet.com/datasheet-pdf/view/517588/TI1/LM35.html

[8] alldatasheet.com. (2014–2021). LM317 pdf, LM317 description, LM317 datasheets, LM317 view ::: ALLDATASHEET ::: [LM317 Datasheet]. Recuperado de https://pdf1.alldatasheet.com/datasheet-pdf/view/22749/STMICROELECTRONICS/LM317.html

[9] puntoflotante. (2017–2021). Sensor infrarrojo de movimiento PIR HC-SR501 [PIR HC-SR501 Datasheet]. Recuperado de https://www.puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf