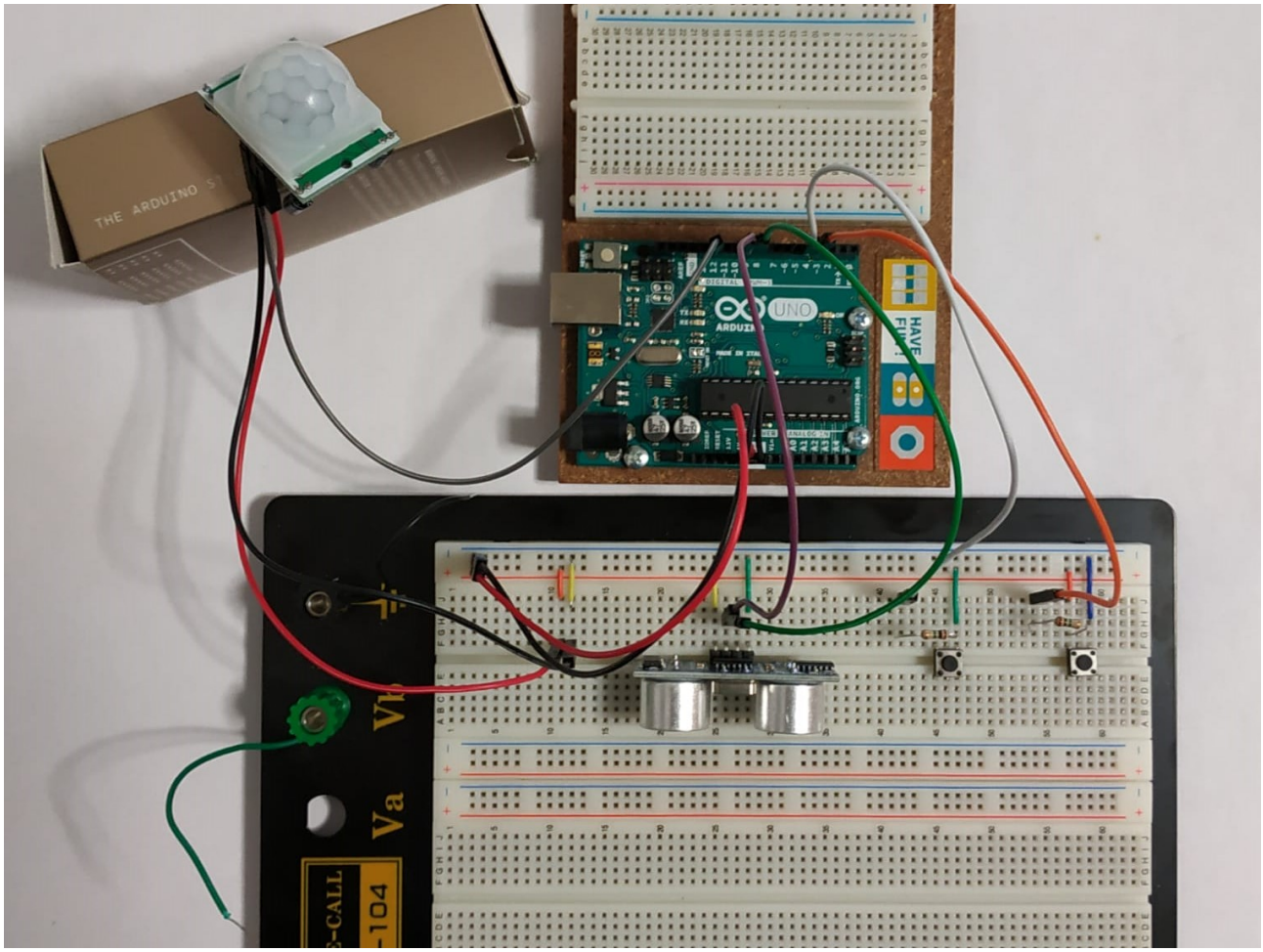
SISTEMA ANTIRROBO



DATOS DE LOS PARTICIPANTES		
NOMBRE	**APELLIDOS**	**NÚMEROS DE MATRÍCULA**
Zeqi	Lin	54694
Martin Augusto	Reigadas Teran	54820
Xiaolong	Ruan	54978

INTRODUCCIÓN

El proyecto se basa en un sistema antirrobo que detecte a un intruso dentro de una cierta área, si en un determinado tiempo el intruso no se aleja, se guardará la fecha y hora en las que estuvo. En el caso de que el intruso fuerza su entrada, se activará la alarma. Y todos los eventos producidos se almacenará en el ordenador conectado al arduino.

OBJETIVOS DEL TRABAJO

- Crear un sistema de seguridad básico.
- Poder detectar objetos en movimiento en una determinada área.
- Asegurarse de que lo que detecte sea un intruso.
- El control de los aparatos a partir de un arduino.
- Programar el arduino para que lea los datos de los sensores.
- Que el arduino envie los datos recibidos al ordenador.
- Que el ordenador guarde los datos.

SENSORES Y ACTUADORES

- Sensor de ultrasonidos
- Sensor infrarrojo pasivo
- Pulsador

DESARROLLO DE SOFTWARE

El arduino lee los datos que recibe de los sensores y según los datos que reciba enviará cierta información al ordenador para su posterior procesamiento.

Para el sensor de movimiento infrarrojo pasivo (PIR):
El sensor de movimiento infrarrojo pasivo envía un uno (o HIGH) si detecta movimiento en su rango y si no detecta nada envía un cero(o LOW).
Si detecta movimiento por más de 30 segundos el arduino envía una señal al ordenador.
Después de ya haber detectado a un intruso si han pasado 30 segundos sin detectar movimiento entonces le enviará esa información al ordenador.

La función que se encarga de este proceso es InfraRedSensor, la función intervalo se encarga de según el valor de una variable global actualizar el tiempo inicial del intervalo o el tiempo actual y luego calcula el desfase de tiempo entre el inicial y el actual.

El tiempo para cuando el **PIR** detecta movimiento es distinto para cuando no lo detecta, para así evitar que el intruso no pueda estar más de los 30 segundos sin que el arduino avise al ordenador.

El sistema para saber si se ha forzado la entrada:
Primero comprueba si se ha abierto la puerta mediante un pulsador, y lo rectifica con un sensor de ultrasonidos que mide la distancia entre la pared del frente del sensor que es contigua a la puerta, así al momento de abrirla el sensor la detecte.

Se fija una diferencia de distancia mínima de 10 cm, para que se active la alarma.
La función que se encarga de calcular la distancia en centímetros es la función ping, y la de pulsador es para medir la diferencia de distancia y compararla con el D_MAX.

```
#define Max_on 30
#define Max_off 30 //en segundos
#define D_MAX 10 // en cm

const int PIR=12;
char flag=0,a=0,h=0;
unsigned long t_i; // tiempo inicial
```

```
const int TriggerPin = 9;
```

```
const int EchoPin =8;
const int PUL_Puerta= 3;
const int PUL_Desactivar=4;
int distancia_o;


void setup() {
Serial.begin(9600);
pinMode(PIR,INPUT);
pinMode(PUL_Puerta,INPUT);
pinMode(TriggerPin,OUTPUT);
pinMode(EchoPin,INPUT);
pinMode(PUL_Desactivar,INPUT);
delay(1000);
distancia_o=ping(TriggerPin,EchoPin);
}

void loop() {
   int pir;
   int lectura;
   lectura=digitalRead(PUL_Puerta);
   pir=digitalRead(PIR);

   InfraRedSensor(pir);


   if(lectura==1 && h==0){
      pulsador(&h);
   }
   if(digitalRead(PUL_Desactivar))
   {
      Serial.print('4');
      h=0;
   }

}


void InfraRedSensor(int pir){
int d_on,d_off;//desfases de tiempo para cuando pir==1 y para pir==0
```

```
if(pir==1){

if(flag==1){
  a=0;
  d_off=0;
  intervalo(&d_on);
}else{
   intervalo(&d_on);
   if(d_on>Max_on){
       Serial.print('2');
       flag=1;
       d_off=0;
   }
}

    }else {

       if(flag==1){
       intervalo(&d_off);

       if(d_off>Max_off){
       Serial.print('3');
       flag=0;
       d_on=0;
       }

        }else{
        a=0;
        intervalo(&d_off);
        d_on=0;
        }
        }

}


void intervalo(int * des){
unsigned long tiempo_actual
if(a==0)
{
   t_i=millis()/1000;
```

```
   *des = 0;
   a=1;
}else{
    tiempo_actual=millis()/1000;
    *des=tiempo_actual-t_i;
   }
}


void pulsador(int* c){
int distancia_f;
int diferencia;

distancia_f = ping(TriggerPin,EchoPin);
diferencia=abs(distancia_f-distancia_o);

if(diferencia > D_MAX){
   Serial.print('1');
   *c=1;
}
*c=0
}


int ping(int TriggerPin,int EchoPin){
long duration, distanceCm;
digitalWrite(TriggerPin,LOW);
delayMicroseconds(4);
digitalWrite(TriggerPin,HIGH);
delayMicroseconds(10);
digitalWrite(TriggerPin,LOW);

duration= pulseIn(EchoPin,HIGH);
distanceCm = duration *10 /292 /2;
return distanceCm;
}
```

CONEXIÓN ARDUINO-C

```
#include "SerialPort.h"
void Crear_Conexion(SerialPort *PuertoSerie, char *portName)
{

PuertoSerie->connected = 0;
PuertoSerie->portName = portName;
PuertoSerie->handler = CreateFileA((portName),
GENERIC_READ | GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,
NULL);
if (PuertoSerie->handler == INVALID_HANDLE_VALUE)
{
if (GetLastError() == ERROR_FILE_NOT_FOUND)
{
printf("ERROR: Handle was not attached. Reason: %s not available\n",
portName);
}
else
{
printf("ERROR!!!");
}
}
else
{
DCB dcbSerialParameters = {0};

if (!GetCommState(PuertoSerie->handler, &dcbSerialParameters))
{
printf("failed to get current serial parameters");
}
else
{
dcbSerialParameters.BaudRate = CBR_9600;
dcbSerialParameters.ByteSize = 8;
dcbSerialParameters.StopBits = ONESTOPBIT;
dcbSerialParameters.Parity = NOPARITY;
dcbSerialParameters.fDtrControl = DTR_CONTROL_ENABLE;
```

```
if (!SetCommState(PuertoSerie->handler, &dcbSerialParameters))
{
printf("ALERT: could not set Serial port parameters\n");
}
else
{
PuertoSerie->connected = 1;
PurgeComm(PuertoSerie->handler, PURGE_RXCLEAR | PURGE_TXCLEAR);
Sleep(ARDUINO_WAIT_TIME);
}
}
}
return ;
}

void CerrarConexion(SerialPort * PuertoSerie)
{
if (PuertoSerie->connected)
{
PuertoSerie->connected = 0;
CloseHandle(PuertoSerie->handler);
}
}

int readSerialPort(SerialPort * PuertoSerie,char *buffer, unsigned int
buf_size)
{
DWORD bytesRead;
unsigned int toRead = 0;

ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status);

if (PuertoSerie->status.cbInQue > 0)
{
if (PuertoSerie->status.cbInQue > buf_size)
{
toRead = buf_size;
}
else toRead = PuertoSerie->status.cbInQue;
}
```

```
memset(buffer, 0, buf_size);

if (ReadFile(PuertoSerie->handler, buffer, toRead, &bytesRead, NULL))
return bytesRead;

return 0;
}

int writeSerialPort(SerialPort *PuertoSerie,char *buffer, unsigned int
buf_size)
{
DWORD bytesSend;

if (!WriteFile(PuertoSerie->handler, (void*) buffer, buf_size,
&bytesSend, 0))
{
ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status);
return 0;
}
else return 1;
}

int isConnected(SerialPort *PuertoSerie)
{
if (!ClearCommError(PuertoSerie->handler, &PuertoSerie->errors,
&PuertoSerie->status))
PuertoSerie->connected = 0;
return PuertoSerie->connected;
}
```

PROGRAMA EN DEVC++

Devc++ recibe los datos enviados por el arduino mediante el puerto serie, y según los datos que reciba guardará la fecha y el tipo de suceso en un fichero binario, para evitar que pueda ser manipulada la información con facilidad.

El fichero es actualizado cada vez que llegué información del arduino, agregando la nueva información a este.
El fichero solo se abrirá al momento de guardar los datos, en ningún otro momento del programa se abrirá, para evitar trabajar con el fichero abierto en todo momento.

Como se abre el fichero con la opción de añadir, entonces si el fichero no existe se creará uno nuevo, y si ya existe se abrirá para agregar la nueva información.
Estos datos podrán ser leídos a posteriori con un programa que maneje archivos binarios (de extensión .bn).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include<time.h>
#include<windows.h>
#include "SerialPort.h"
#define MAX_DATA_LENGTH 255

typedef struct{
      char fecha[26];
      char dato[30];
}Recep;

/*Funciones prototipo*/
void fecha(char[]);
void lectura(char*);
void guardar_datosbn(Recep *);
void autoConnect(SerialPort *arduino,char*);


int main(void)
{
 /*Arduino SerialPort object*/
 SerialPort *arduino;
 /* Puerto serie en el que está Arduino*/
 char* portName = "COM7";
 /* Buffer para datos procedentes de Arduino*/
 char incomingData[MAX_DATA_LENGTH];

 /* Crear estructura de datos del puerto serie*/
 arduino = (SerialPort *)malloc(sizeof(SerialPort));
 /* Apertura del puerto serie*/
 Crear_Conexion(arduino,portName);
 autoConnect(arduino,incomingData);

}


void autoConnect(SerialPort *arduino,char *incomingData)
{
 int readResult;
```

```
/* Espera la conexión con Arduino*/
while (!isConnected(arduino))
{
Sleep(100);
Crear_Conexion(arduino,arduino->portName);
}
 /*Comprueba si arduino está connectado*/
if (isConnected(arduino))
{
printf ("Conectado con Arduino en el puerto %s\n",arduino->portName);
}
 /*Bucle de la aplicación*/
while (isConnected(arduino))
{
 readResult = readSerialPort(arduino,incomingData,MAX_DATA_LENGTH);
if (readResult!=0){
      printf("%s\n", incomingData);
      lectura(incomingData);
}
  Sleep(10);

}
if (!isConnected(arduino))
 printf ("Se ha perdido la conexión con Arduino\n");
}


void guardar_datosbn(Recep *r){
            int i;
      FILE *salida;
      salida = fopen("Info.bn","ab");

      if(salida==NULL)
            printf("No se ha podido guardar los datos");
      else{

      fwrite(r,sizeof(Recep),1,salida);

            fclose(salida);
}
}

void lectura(char *d){
      Recep info;
      char Intruso[]="Intruso en la zona";
      char Activa[]="Alarma Activada";
      char NoIntruso[]="Intruso fuera de la zona";
      char Desactivar[]="Alarma Desactivada";
```

```
      switch(d[0]){
            case '1':
                  fecha(info.fecha);
                  strcpy(info.dato,Activa);
                  break;
            case '2':
                  fecha(info.fecha);
                  strcpy(info.dato,Intruso);
                  break;
            case '3':
                  fecha(info.fecha);
                  strcpy(info.dato,NoIntruso);
                  break;
            case '4':
                  fecha(info.fecha);
                  strcpy(info.dato,Desactivar);
                  break;

      }

      guardar_datosbn(&info);

}



void fecha(char fecha[]){
            time_t current_time;
    current_time=time(NULL);
    ctime(&current_time);
    strcpy(fecha, ctime(&current_time));

}
```