Alcoa Young Talent Competition II

Panel de entrenamiento modular para la automatización y control de procesos

Módulo de entrenamiento: Portón corredizo

Desarrollado por Sergio Sanjurjo Montero

1º Bachillerato A

Colegio Corazón de María (Gijón)

Alcoa Foundation

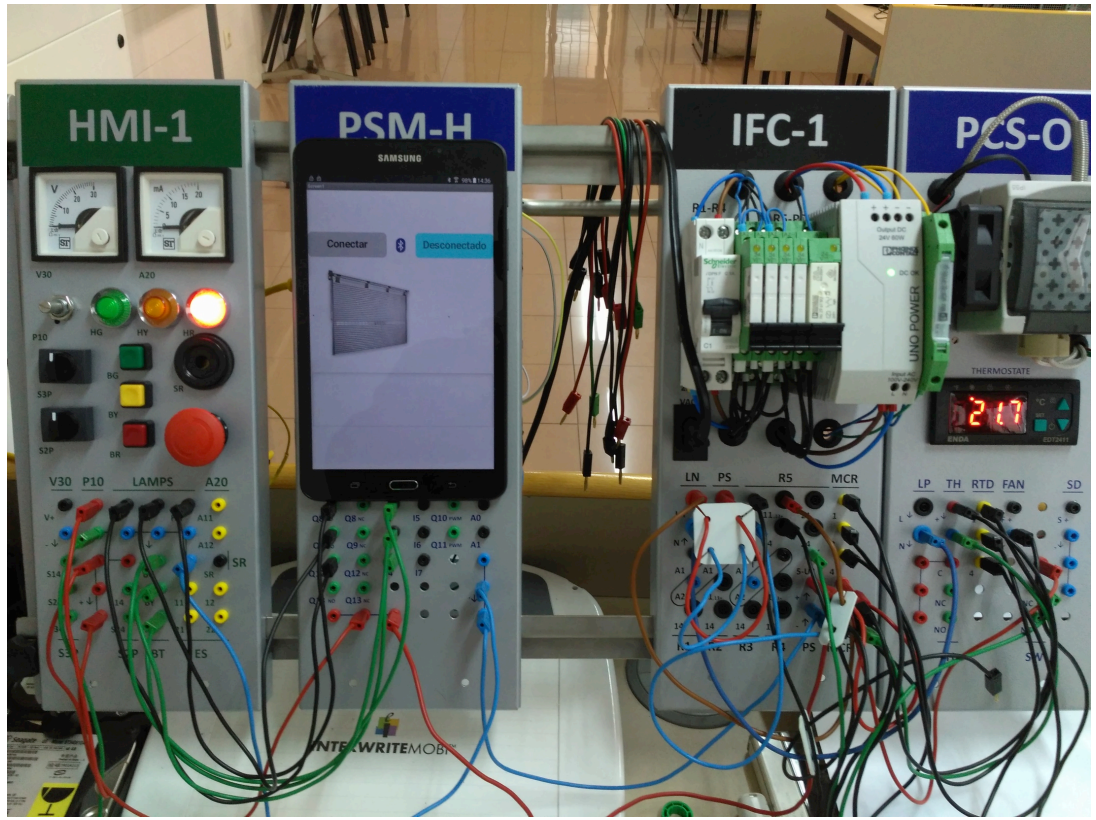
Universidad de Oviedo EPI Gijón

Claretianos Gijón

PRÁCTICAS DE AUTOMATIZACIÓN Y CONTROL
MÓDULOS DE PANEL DE ENTRENAMIENTO

Objetivos

- Proyecto de automatización (Portón)
- Control mediante Arduino UNO
- Señales E/S digitales
- Cableado de señales de mando y fuerza
- Programación en IDE Arduino.
- Control mediante placa Arduino.
- Programación de aplicación para Smartphone Android con AppInventor.
- Descarga, depuración y puesta en marcha.



Hardware y Software

Modulo entrenador en automatización:

- Módulo HMI-1: Interface de usuario.
- Módulo IFC-1: Interface de señales (Fuente de alimentación).
- Módulo PSM-H: Incluye Arduino UNO, placa interface de señal y placa con 4 relés para gestión de salidas.
- Smartphone Android

Software IDE Arduino.
AppInventor

Hardware y Software

Modulo HMI-1: Interface de usuario

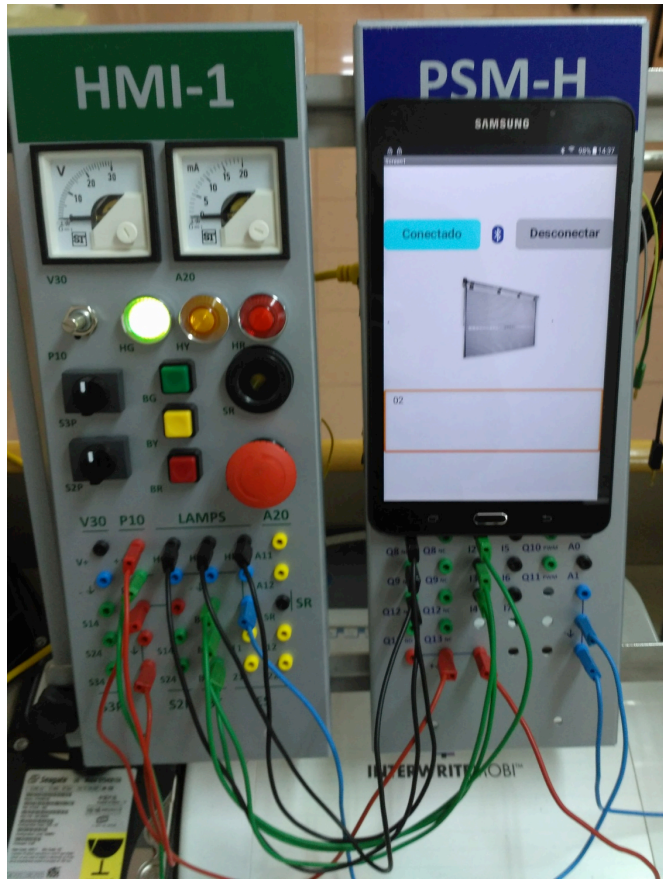
- Se utilizan los pulsadores: Verde (apertura), Rojo (cierre) y Ámbar (emergencia).

- Se utilizan los pilotos lumimosos: Verde (portón abierto), Rojo (portón cerrado) y Ámbar (portón en movimiento o detenido por señal de emergencia)

Hardware y Software

Modulo PSM-H: Control de proceso

- Se conectan los puertos con los pulsadores
- Se conectan los puertos con los pilotos luminosos

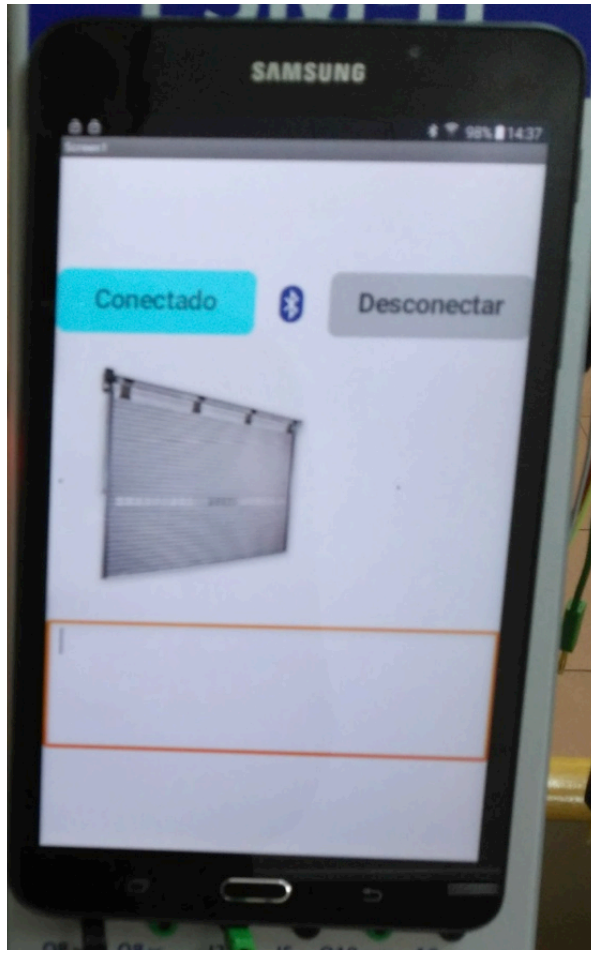


Hardware y Software

Smartphone Arduino

- Posee el software de simulación del proceso.

- Simula un portón corredizo motorizado con dos finales de carrera y un sensor de ultrasonidos para la detección de obstáculos durante el cierre del portón.



Desarrollo

Se trata de simular el funcionamiento de una portón mediante la conexión Bluetooth entre un Arduino y una aplicación realizada con App Inventor. Las funciones simuladas son las siguientes:

1)Apertura (Botón “Verde” en el módulo de Interface). Una pulsación comunica que la puerta se abra.

2)Cerrado (Botón “Rojo” en el módulo de Interface). Una pulsación comunica que se cierre la puerta.

3)Parada de emergencia (Seta de emergencia en el módulo de Interface). Una pulsación comunica que se detenga la puerta.

4)Modo de funcionamiento (Conmutador de 2 posiciones en el módulo de Interface). Una posición comunica modo Manual, la otra posición comunica modo automático.

5)Sensor de presencia (Detector de pulsación en la aplicación en el recorrido de la puerta). Durante el cerrado se comprueba si existe algún obstáculo interponiéndose, en cuyo caso la puerta se detiene automáticamente. Cuando el obstáculo es retirado el cerrado continúa automáticamente.

6)Pilotos luminosos (Rojo, ámbar y verde). Se enciende el rojo cuando la puerta está cerrada, el verde cuando está abierta y el ámbar durante el proceso de apertura o cierre.

Comunicación entre Arduino y Smartphone

El sistema de control y el sistema de simulación se comunican vía Bluetooth enviando diversos código de estado del sistema:

1)Se envía el código ‘0’ a la App del Smartphone para indicar que se ha pulsado el botón Verde y se debe abrir el portón. Se envía el código ‘0’ al programa de control de Arduino para indicar que el portón ha alcanzado el final de carrera (apertura).

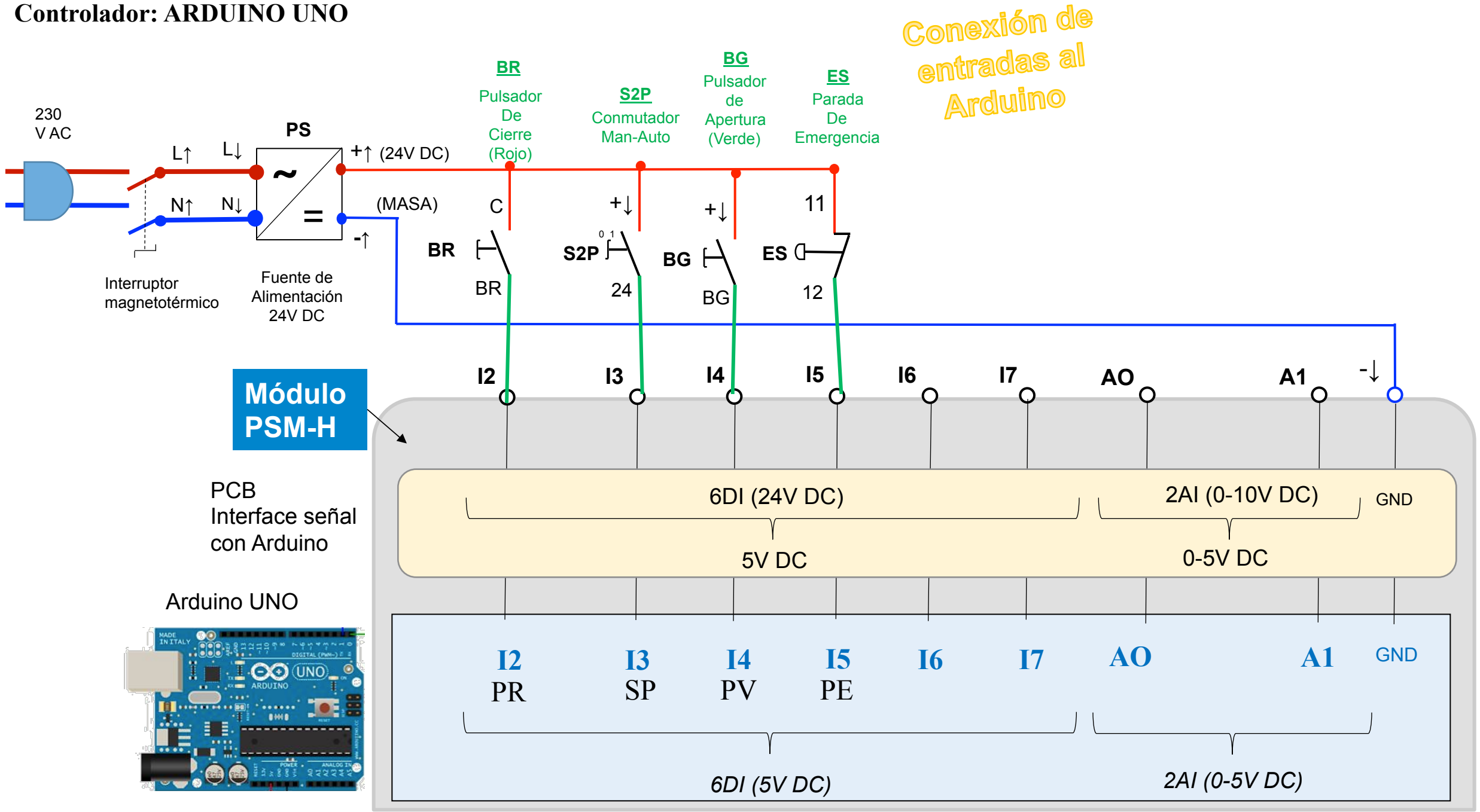
2)Se envía el código ‘1’ a la App del Smartphone para indicar que se ha pulsado el botón Rojo y se debe cerrar el portón. Se envía el código ‘1’ al programa de control de Arduino para indicar que el portón ha alcanzado el final de carrera (cierre).

3)Se envía el código ‘2’ a la App del Smartphone para indicar que se debe detener el portón. Se ha recibido información de: algún final de carrera, sensor de ultrasonidos o pulsación de la seta de emergencia. Se envía el código ‘2’ al programa de control de Arduino para indicar que el sensor de ultrasonidos (toque de la pantalla) ha detectado un obstáculo durante el cierre del portón.

4)Se envía el código ‘M’ a la App del Smartphone para indicar que el conmutador de modo de funcionamiento está en Manual.

5)Se envía el código ‘A’ a la App del Smartphone para indicar que el conmutador de modo de funcionamiento está en Automático.

Controlador: ARDUINO UNO



Controlador ARDUINO UNO

Módulo PSM-H

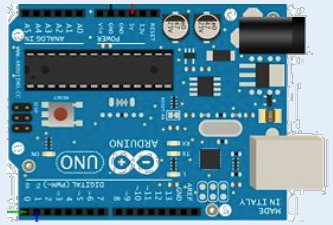
Conexión de salidas al Arduino

Módulo 4 relés

4DO (5V DC)

LR LA LV

+5V Q8 Q9 Q12 Q13 GND



ARDUINO UNO

PWM PWM

Q10 Q11 GND

VCC IN4 IN3 IN2 IN1 GND

4DO (5V DC)

4DO RELÉ

R1 R2 R3 R4

C

2DO PWM 5V DC)

GND

Circuito RC

2AO (0-5V DC) GND

PCB Interface señal con Arduino

+↑ Q8,NO Q9,NO Q12,NO Q13,NO

Q10 PWM Q11 PWM -↓

230 V AC

L↑ L↓ N↑ N↓

PS

~ =

+↑ 24V DC -↑

HR HY HG

-↓ -↓ -↓

Interruptor magnetotérmico

Fuente de Alimentación 24V DC

Luz roja **HR**

Luz ámbar **HY**

Luz verde **HG**

```
// Variables del controlador
unsigned long tiempo = 0;       // Variable auxiliar para realizar pausas
unsigned long cuenta_atras = 0; // Variable auxiliar para realizar una espera
int mov = 0;                    // Portón en movimiento (1) o detenido (0)
int pos = 0;                    // Situación del portón: cerrado (0), intermedia (1), abierto (2)
int ambar = 0;                  // Estado del led ámbar
char dato;                      // Dato enviado desde el simulador
char modo;                      // Modo de cierre: automático (A), manual (M)
const int LR = 8, LA = 9, LV = 12, PR = 2, S2P = 3, PV = 4, PE = 5; // Puertos de los led de señalización y pulsadores

// Incialización del controlador
void setup() {
  Serial.begin(9600);
  pinMode(LR, OUTPUT);
  digitalWrite(LR, HIGH);
  pinMode(LA, OUTPUT);
  pinMode(LV, OUTPUT);
  pinMode(PR, INPUT);
  pinMode(PV, INPUT);
  pinMode(PE, INPUT);
  pinMode(S2P, INPUT);

  if (digitalRead(S2P) == LOW) {
    modo = 'M';
  }
  else {
    modo = 'A';
  }
}

// Procesos de controlador Lectura de sensores y actualización del estado del sistema
void loop(){
  Leer();
  Pulsador();
  Procesa();
}
```

```
// Lectura de sensores del proceso: datos enviados desde el simulador del proceso (finales de carrera y sensor de ultrasonidos)
// Modificación de los estados del proceso según lectura anterior y según el estado actual del sistema
// Procesamiento y actualización de elementos del interface de usuario
//
//  Datos enviados desde el simulador
//  dato = 0      --->     Final de carrera (portón abierto)
//  dato = 1      --->     Final de carrera (portón cerrado)
//  dato = 2      --->     Sensor de ultrasonidos (un obstaculo ha cruzado mientras se cerraba el portón)
//  dato = A      --->     Modo automático
//  dato = M      --->     Modo manual

void Leer() {
  if (Serial.available() > 0){
    dato = Serial.read();
    if (dato == '0'){
      Serial.write('2');      // Se envía la señal de detención del motor
      digitalWrite(LR, LOW);  // Se apaga el led rojo
      digitalWrite(LV, HIGH); // Se enciende el led verde para indicar que el portón está abierto
      digitalWrite(LA, LOW);  // Se apaga el led ámbar
      ambar = 0;
      mov = 0;
      pos = 2;
      if (modo == 'A') {
          cuenta_atras = 20;
      }
    }
    else if (dato == '1'){
      Serial.write('2');      // Se envía la señal de detención del motor
      digitalWrite(LR, HIGH); // Se enciende el led rojo para indicar que el portón está cerrado
      digitalWrite(LV, LOW);  // Se apaga el led verde
      digitalWrite(LA, LOW);  // Se apaga el led ámbar
      ambar = 0;
      mov = 0;
      pos = 0;
    }
```

```
    else if (dato == '2'){
      Serial.write('2');      // Se envía la señal de detención del motor
      digitalWrite(LR, HIGH); // Se enciende el led rojo para indicar que el portón está detenido en medio por un obstaculo
      digitalWrite(LV, HIGH); // Se enciende el led verde para indicar que el portón está detenido en medio por un obstaculo
      digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está detenido en medio por un obstaculo
      ambar = 1;
      mov = 0;
      pos = 1;
      if (modo == 'A') {
          cuenta_atras = 20;
      }
    }
/*
 else if (dato == '3'){
      Serial.write('1');
      digitalWrite(LR, LOW);
      digitalWrite(LV, LOW);
      digitalWrite(LA, HIGH);
      ambar = 1;
      mov = 0;
      pos = 1;
    }
 */
  }
}

// Lectura de sensores del proceso: pulsadores
// Modificación de los estados del proceso según lectura anterior y estado actual del sistema
// Procesamiento y actualización de elementos del interface de usuario
//
//  Estados enviados hacia el simulador
//  dato = 0      --->     Pulsador Verde (Abrir el portón)
//  dato = 1      --->     Pulsador Rojo (Cerrar el portón)
//  dato = 2      --->     Pulsador de emergencia (Detener el portón)
//  dato = A      --->     Conmutador de modo de cierre (Automático)
//  dato = M      --->     Conmutador de modo de cierre (Manual)
```

```
void Pulsador() {
  if (digitalRead(PV) == HIGH && mov == 0 && pos != 2) {
    Serial.write('0');      // Se envía la señal de arranque del motor para apertura del portón
    digitalWrite(LR, LOW);  // Se apaga el led rojo
    digitalWrite(LV, LOW);  // Se apaga el led verde
    digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está en movimiento
    ambar = 1;
    mov = 1;
    pos = 1;
    cuenta_atras = 0;
  }
  else if (digitalRead(PR) == HIGH && mov == 0 && pos != 0) {
    Serial.write('1');      // Se envía la señal de arranque del motor para cierre del portón
    digitalWrite(LR, LOW);  // Se apaga el led rojo
    digitalWrite(LV, LOW);  // Se apaga el led verde
    digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está en movimiento
    ambar = 1;
    mov = 1;
    pos = 1;
    cuenta_atras = 0;
  }
  else if (digitalRead(S2P) == HIGH && modo == 'M') {
    Serial.write('A');      // Se envía la señal de modo de cierre automático
    modo = 'A';
  }
  else if (digitalRead(S2P) == LOW && modo == 'A') {
    Serial.write('M');      // Se envía la señal de modo de cierre manual
    modo = 'M';
  }
```

```
  else if (digitalRead(PE) == HIGH) {
    if (mov == 1) {
      Serial.write('2');      // Se envía la señal de detención del motor porque se ha producido una emergencia
      digitalWrite(LR, HIGH); // Se enciende el led rojo para indicar que el portón está detenido por emergencia
      digitalWrite(LV, HIGH); // Se enciende el led verde para indicar que el portón está detenido por emergencia
      digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está detenido por emergencia
      ambar = 1;
      mov = 0;
      pos = 1;
      if (modo == 'A') {
          cuenta_atras = 20;
      }
    }
    else if (pos == 1) {
        Serial.write('0');      // Se envía la señal de arranque del motor tras la parada de emergencia
        digitalWrite(LR, LOW);  // Se apaga el led rojo
        digitalWrite(LV, LOW);  // Se apaga el led verde
        digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está en movimiento
        ambar = 1;
        mov = 1;
        pos = 1;
        cuenta_atras = 0;
    }
    Espera(500);
  }
}



void Espera (unsigned long pausa) {
  tiempo = millis();
  while ( millis () < tiempo + pausa ) Leer();
}
```

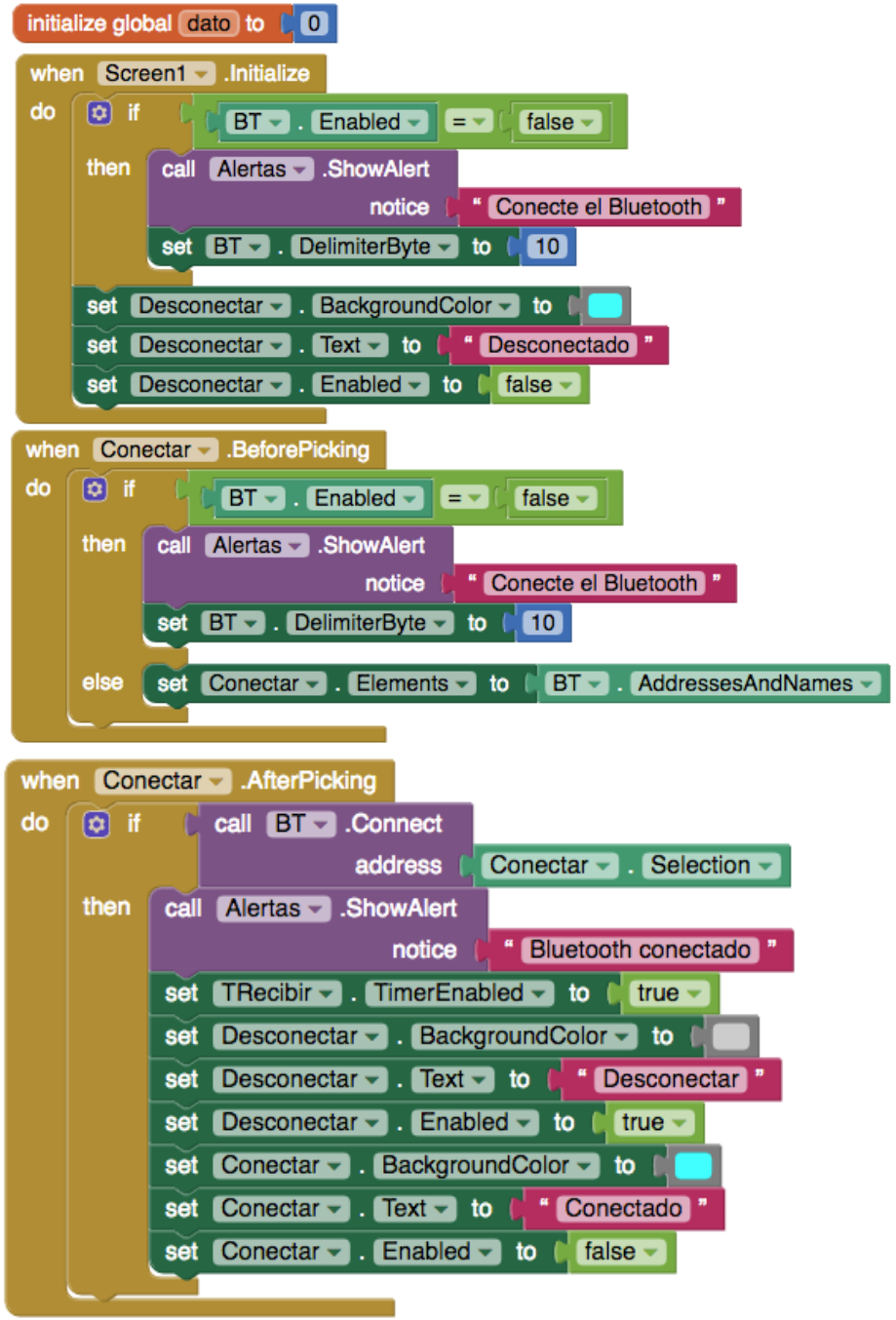
```
void Procesa() {
  if (modo == 'A' && pos > 0) {
    if (cuenta_atras > 0) {
        cuenta_atras = cuenta_atras - 1;
        Espera(250);
    }
    else if (mov == 0 && pos > 0) {
        if (pos == 1) {
          Serial.write('0');    // Se envía la señal de arranque del motor para apertura del portón
        }                       // porque se detuvo por emergencia y parado en mitad del recorrido
        else {
           Serial.write('1');   // Se envía la señal de arranque del motor para cierre del portón
        }                       // porque el portón está abierto y termino la espera (cuenta_atras)
        digitalWrite(LR, LOW);  // Se apaga el led rojo
        digitalWrite(LV, LOW);  // Se apaga el led verde
        digitalWrite(LA, HIGH); // Se enciende el led ámbar para indicar que el portón está en movimiento
        ambar = 1;
        mov = 1;
        pos = 1;
    }
  }
  if (cuenta_atras == 0 && pos == 1) {
    if (mov == 1) {
      digitalWrite(LA, ambar);
      ambar = !ambar;
      Espera(250);
    }
/*
  if (mov == 0) {
        digitalWrite(LR, ambar);
        digitalWrite(LV, ambar);
        digitalWrite(LA, ambar);
    }
*/
  }
}
```

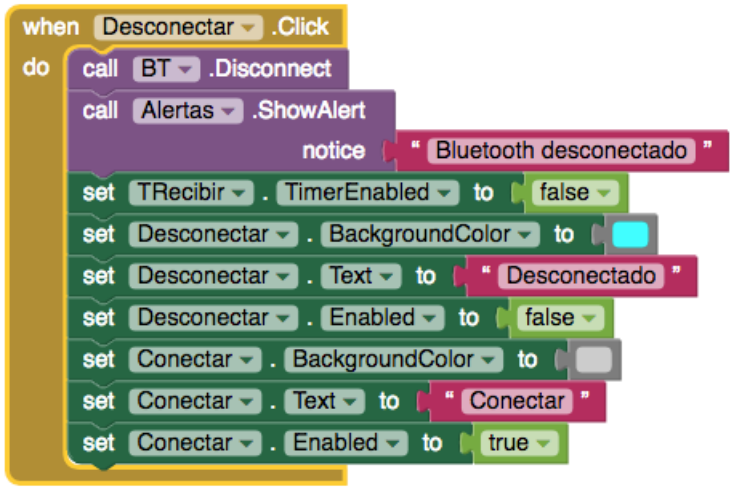
Conectar

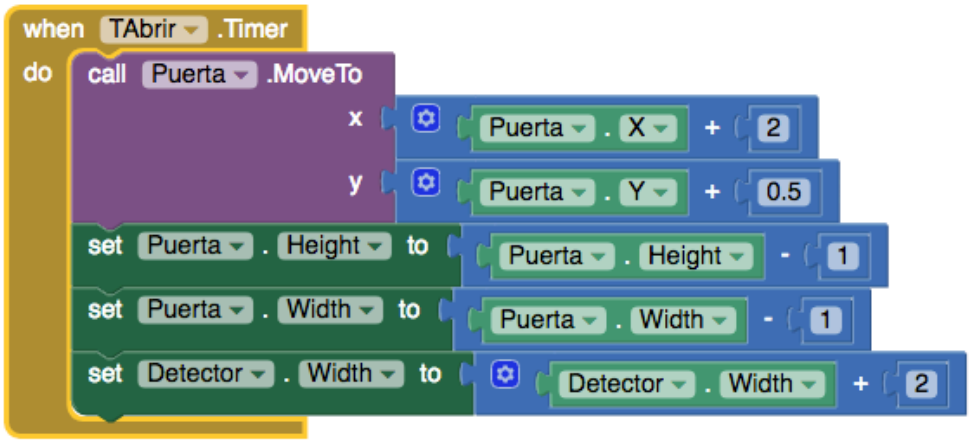
Desconectado

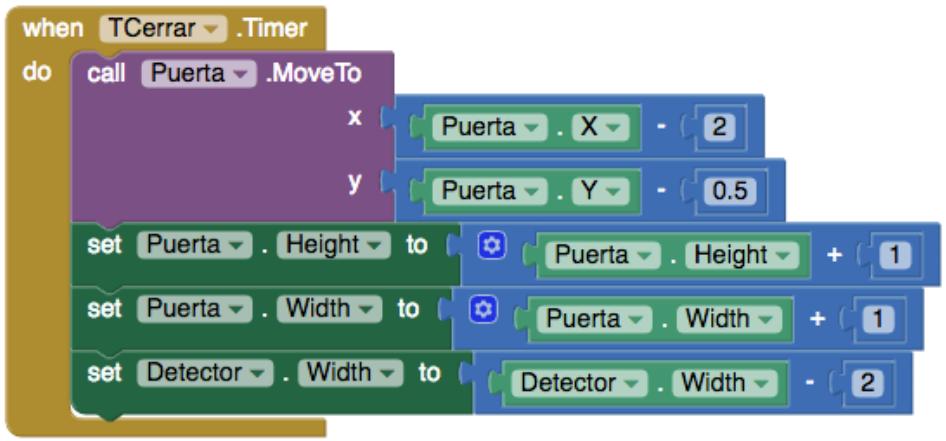
Automático

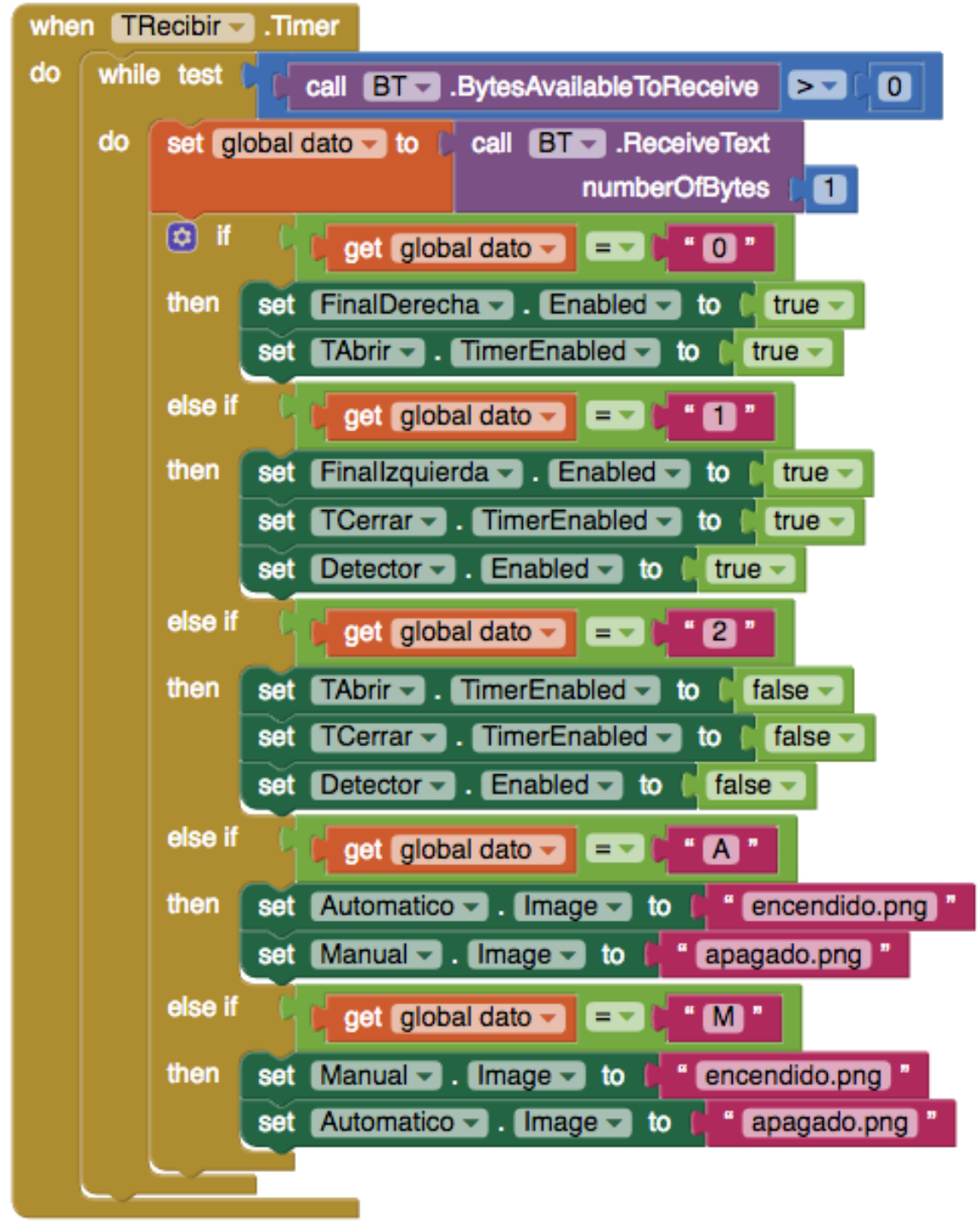
Manual

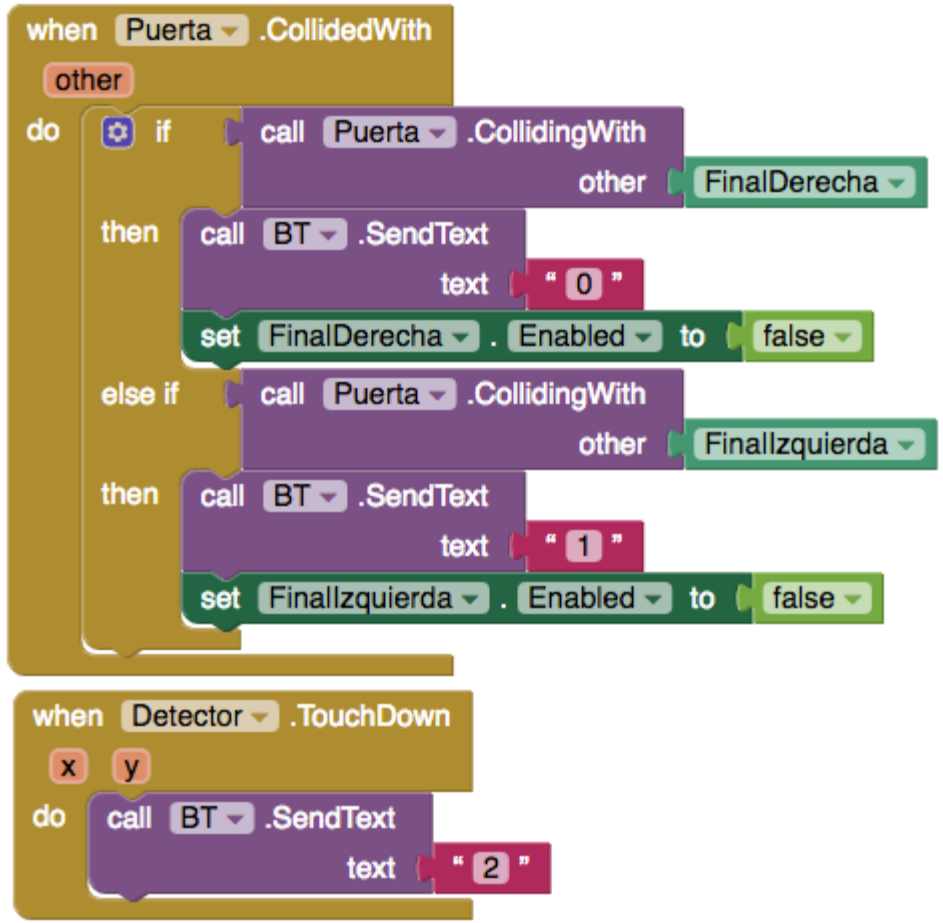



when Desconectar .Click
do call BT .Disconnect
call Alertas .ShowAlert
notice " Bluetooth desconectado "
set TRecibir . TimerEnabled to false
set Desconectar . BackgroundColor to
set Desconectar . Text to " Desconectado "
set Desconectar . Enabled to false
set Conectar . BackgroundColor to
set Conectar . Text to " Conectar "
set Conectar . Enabled to true


when TAbrir .Timer
do call Puerta .MoveTo
x Puerta . X + 2
y Puerta . Y + 0.5
set Puerta . Height to Puerta . Height - 1
set Puerta . Width to Puerta . Width - 1
set Detector . Width to Detector . Width + 2


when TCerrar .Timer
do call Puerta .MoveTo
x Puerta . X - 2
y Puerta . Y - 0.5
set Puerta . Height to Puerta . Height + 1
set Puerta . Width to Puerta . Width + 1
set Detector . Width to Detector . Width - 2


when TRecibir .Timer
do while test call BT .BytesAvailableToReceive > 0
do set global dato to call BT .ReceiveText
numberOfBytes 1
if get global dato = " 0 "
then set FinalDerecha . Enabled to true
set TAbrir . TimerEnabled to true
else if get global dato = " 1 "
then set FinalIzquierda . Enabled to true
set TCerrar . TimerEnabled to true
set Detector . Enabled to true
else if get global dato = " 2 "
then set TAbrir . TimerEnabled to false
set TCerrar . TimerEnabled to false
set Detector . Enabled to false
else if get global dato = " A "
then set Automatico . Image to " encendido.png "
set Manual . Image to " apagado.png "
else if get global dato = " M "
then set Manual . Image to " encendido.png "
set Automatico . Image to " apagado.png "


when Puerta .CollidedWith
other
do if call Puerta .CollidingWith
other FinalDerecha
then call BT .SendText
text " 0 "
set FinalDerecha . Enabled to false
else if call Puerta .CollidingWith
other FinalIzquierda
then call BT .SendText
text " 1 "
set FinalIzquierda . Enabled to false
when Detector .TouchDown
x y
do call BT .SendText
text " 2 "