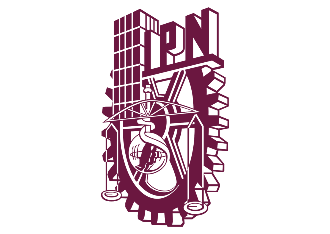
**INSTITUTO POLITÉCNICO NACIONAL**

UNIDAD INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Proyecto Final

UGV controlado remotamente por bluetooth y GUI-Matlab

**Microprocesadores, microcontroladores e interfaz**

**Integrantes**

Conde Jaimes Jorge Andrés

Espinoza Reyes Ulises Gubish

**Docente:**

León Ponce Moisés

2MM8

Introducción

En la actualidad los microprocesadores y los microcontroladores tienen un gran impacto en el desarrollo de máquinas y dispositivos que día a día usamos, y que todos tenemos en nuestras casas para poder realizar actividades específicas como lo son al usar electrodomésticos para calentar o hacer de comer, para ver televisión o escuchar radio, incluso para que el automóvil encienda o para que las llamadas telefónicas puedan efectuarse. En cualquier parte que veamos ya sea en la casa, en la escuela, en el transporte público, etc. podemos notar que hay una infinidad de dispositivos que los usan para poder efectuar su función de manera precisa y eficaz.

Hoy en día existen muchos carros a control remoto (también se les llama vehículos no tripulados) que se suelen usar para el entretenimiento, pero también hay carros que son utilizados para realizar alguna actividad en la que puedan ofrecer un gran desempeño al efectuarlo de buena forma. Algunos usos de estos en la actualidad son en el ámbito militar, la inspección y vigilancia de líneas eléctricas y de tuberías, el patrullaje de fronteras, las misiones de rescate, el monitoreo de regiones, la búsqueda de petróleo y gas natural, la prevención de incendios, la topografía y desastres naturales, la agricultura, la entrega de paquetes, etc. Además, estos vehículos pueden ser usados en algunas situaciones que incluyan algún riesgo de seguridad para los seres humanos.

Los vehículos no tripulados son herramientas que no llevan piloto o son controladas remotamente, estos pueden portar cámaras, sensores e incluso armas para así realizar la actividad para la cual fue elaborado. Se clasifican en 4 tipos:

* Vehículos submarinos no tripulados (Unmanned Underwater Vehicles, UUV).
* Vehículos marinos no tripulados (Unmanned Surface Vehicles, USV).
* Vehículos aéreos no tripulados (Unmanned Aerial Vehicles, UAV).
* Vehículos terrestres no tripulados (Unmanned Ground Vehicles, UGV).

Un vehículo terrestre no tripulado (UGV) es un vehículo operando en contacto con la tierra sin presencia humana a bordo. Existen dos tipos de UGV: los operados remotamente y los vehículos autónomos. Un UGV de operación remota es aquel que realiza acciones con base en las instrucciones dadas por un operador a través de un enlace de comunicación. Todos los movimientos son determinados por el operador de acuerdo con información sensorial de entrada como puede ser observación en línea de visión o video cámaras digitales. Por otra parte, un vehículo autónomo utiliza inteligencia artificial (IA) en todo su curso de acciones.

El control remoto de los vehículos autónomos operados remotamente puede llevarse a cabo vía infrarrojo, radiofrecuencia RF (bluetooth, Wi-Fi) o tecnología GSM. Todo va a depender de las limitaciones y las necesidades que se tengan al plantear el proyecto.

Resumen

La programación e implementación de microcontroladores en dispositivos que ayuden a realizar ciertos trabajos o actividades se ha vuelto más común hoy en día. Un ejemplo de estos son el desarrollo de los *vehículos terrestres no tripulados (UGV)* para la realización de diversas actividades dentro de procesos, exploración o en actividades que pueden ser peligrosas para un humano.

En este proyecto se elaboró un prototipo de UGV controlado vía remota por bluetooth desde una *interfaz gráfica de usuario (GUI)* desarrollada en el software Matlab. Para la construcción del prototipo se desarrollo un programa en lenguaje c que controla todo el funcionamiento a través del pic16f887 y que, a la vez, este recibe la información de que debe hacer a través de la GUI elaborada en *Matlab* mediante conexión bluetooth de tal manera que el usuario pueda escoger el modo (demostración, medición, seguidor de líneas y control manual) y la velocidad (lento, regular y rápido) del UGV. El prototipo fue elaborado con partes recicladas de *MDF* de un proyecto viejo, al igual de placas fenólicas recubiertas de cobre para el *PCB (placa de circuito impreso)* diseñado en el software Eagle de autodesk y los componentes electrónicos descritos en la tabla de componentes.

Palabras clave

Vehículos terrestres no tripulados (UGV), interfaz grafica de usuario (GUI), bluetooth, PCB (placa de circuito impreso) y MDF.

Objetivo

Demostrar que los conocimientos adquiridos en la materia de microprocesadores, microcontroladores e interfaz son satisfactorios al momento de aplicarlos en un proyecto que requiera la programación de PICs, su comunicación con la interfaz gráfica de MATLAB, la implementación de sensores como el ultrasónico (distancia) o los seguidores de línea y como extra la elaboración de PCB del circuito.

Componentes

A continuación, se presentaran los componentes y materiales que se emplearon en la elaboración del proyecto junto con una descripción y las características, al igual que la cantidad de cuantos de cada uno se utilizaron y su respectivo costo en el mercado.

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Cantidad | Costo (MXP $) | Imagen |
| PIC16f887 | 1 | $150 |  |
| Sensor ultrasónico  [HC-SR04] | 1 | $49 | Ultrasonic Distance Sensor - HC-SR04 - SEN-15569 - SparkFun Electronics |
| Seguidor de línea  [KY-033] | 2 | $39 | Sensor de Obstáculos Módulo KY-033 - UNIT Electronics |
| Base para PIC torneada de 40 pines | 1 | $6 | Base Torneada para Integrado DIP 40 Pines | MCI Electronics.cl |
| Motorreductor de 1 eje | 2 | $49 |  |
| Regulador de voltaje [LM7805] | 1 | $12 |  |
| Modulo bluetooth  [HC-06] | 1 | $200 | Modulo Bluetooth HC-06 |
| Modulo puente H  [L293D] | 1 | $30 | 5x Circuito Puente H Driver L293d L293 Control De Motores | MercadoLibre |
| Capacitor electrolítico | 2 | $3 |  |
| Pines | 36 | $5 |  |
| Baterías de polímetro de litio | 2 | $90 |  |
| Base para circuito integrado de 16 pines | 1 | $4 |  |
| Placa fenólica de 5x10 | 1 | $10 | Placa fenólica una cara 5 x 10 cm - Geek Factory |
| Cables Dupont | 10 | $29 |  |
| Llanta para carrito | 2 | Ya en posesión. |  |

Funcionamiento

El funcionamiento del programa se basa en 4 tipos de funciones que son seguidor de líneas, rutina de movimiento, control manual y medición de distancia. Requerimos funciones de comunicación USART para poder operar nuestro PIC con éxito. De modo que definimos las siguientes funciones:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

De modo que requeriremos un modulo Bluetooth para la comunicación. En este caso utilizamos el HC-06, Módulo Bluetooth HC-06 es un dispositivo que soporta conexiones inalámbricas a través del protocolo “bluetooth”. Si algún dispositivo se conecta, el módulo transmite a este todos los datos que recibe desde nuestro microcontrolador y viceversa.

El módulo bluetooth HC-06 dispone de 4 pines y solo puede actuar como esclavo y además dispone de un juego reducido de instrucciones a las que atiende. Permite una conexión sencilla y sin problemas mediante comandos AT a través de una puerta serie. Permite la comunicación de voz y datos a través de una red inalámbrica llamada WPAN (Wireless Personal Area Network) por sus siglas en ingles de Red de Área Personal Inalámbrica

Los módulos HC-06 están montados sobre una interfaz en la que se incluye cuatro pines para su conexión, Vcc, Gnd, Txd y Rxd, además cuentan con un led para indicar el estado de conexión del bluetooth. Si este led parpadea es que no está emparejado, si este está activado de forma continua el bluetooth esta emparejado.

Diagrama, Esquemático

Descripción generada automáticamente con confianza media

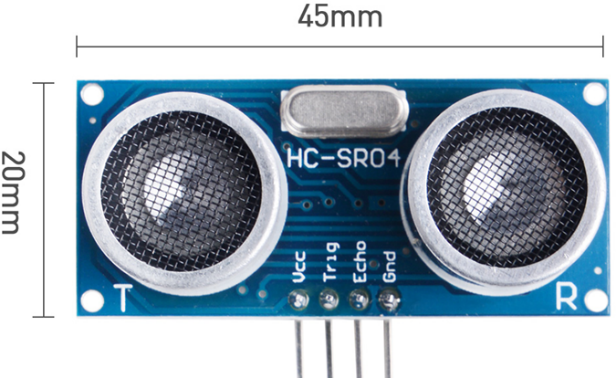
Dentro de nuestro programa tenemos una selección de modo que funciona con una sentencia switch-case donde el caso es obtenido a partir de un dato de la terminal con nuestra función getch() mostrada para obtener datos.

Como primera opción nuestro programa obtiene una medición por medio de un módulo ultrasónico HC-SR04, de la siguiente manera:

Interfaz de usuario gráfica

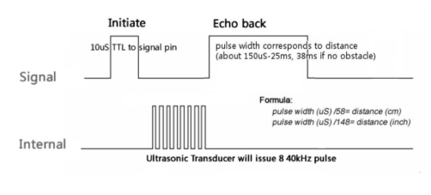
Descripción generada automáticamente con confianza media

La funcionalidad del SR-HC04 se basa en que incorpora un par de transductores de ultrasonido que se utilizan de manera conjunta para determinar la distancia del sensor con un objeto colocado enfrente de este. Un transductor emite una “ráfaga” de ultrasonido y el otro capta el rebote de dicha onda. El tiempo que tarda la onda sonora en ir y regresar a un objeto puede utilizarse para conocer la distancia que existe entre el origen del sonido y el objeto.



Como se puede observar, el HC-SR04 genera un pulso en el pin marcado como “echo” cuya duración es proporcional a la distancia medida por el sensor.

Para obtener la distancia en centímetros, solamente debemos dividir el tiempo en microsegundos entre 58 o para obtener la distancia en centímetros (148 para pulgadas).



Después tendrá una función de rutina de movimiento, la cual realizará movimientos con motores predefinidos dentro del PIC de la siguiente manera:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Esta función requerirá de un integrado para el control de motores, en este caso usaremos el L293D el cual es un controlador de motores. Esta construido con 4 mitades de puente-H.

El L293D está construido con partes de puentes-H, como ya se había mencionado. Está característica nos permite controlar distintas cantidades de motores. Por ejemplo, puede controlar 4 motores de DC unidireccionalmente o 2 motores (DC) en ambas direcciones. También podría controlador un motor a pasos de bipolar. Como puedes ver, tiene varias aplicaciones para diferentes motores.

Una de las principales ventajas del controlador, es que permite una alimentación independiente para los motores. Por ejemplo, se pueden controlar motores desde los 4.5 VDC hasta 36 VDC. Es importante mencionar, que para cuando se manejan potencias mayores a 5 watts, se necesita un buen disipador. El voltaje lógico de control es de 5 VDC. A pesar de esta restricción, podría ser controlado con lógica de 3.3 V. La desventaja de usar lógica de 3.3 V es que aún requeriría una fuente de 5 VDC conectada al pin 16.

Diagrama, Esquemático

Descripción generada automáticamente

Después la opción de seguidor de líneas, utilizara sensores seguidores de líneas los cuales transmitirán datos hacia el PIC para que así puede seguir una trayectoria sin salirse del comino de la siguiente manera:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Utilizando de la misma manera el integrado de motores para su control.

Y finalmente realizará la opción de control manual, en la cual se deberá de seleccionar la dirección en la que avanzara, así como el nivel de velocidad que este tendrá, donde lo aplicamos de la siguiente manera:

Texto

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica

Descripción generada automáticamente

Para la comunicación del PIC con la interfaz gráfica de usuario (GUI) de Matlab primero lo que se hizo fue el diseño de esta. Dentro del software MATLAB, en la ventana de comandos se escribe “GUIDE” para inicializar el editor de la interfaz así generando un archivo. fig en el cual editamos, movimos y la ajustamos los componentes para darle la forma y el formato que requeríamos según las especificaciones de las acciones y procesos que el UVG debía realizar y que la GUI debía comunicarle.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamenteCon la ayuda de las herramientas que se encuentran a la izquierda de la ventana agregamos botones, botones radiales, texto estático, paneles y axis con los cuales le dimos el formato a la interfaz de acuerdo con las especificaciones que el proyecto tenia y que el prototipo debía hacer.

Los axis fueron utilizados para agregar imágenes y que la GUI tuviera una mejor presentación y que no se viera con colores tan sólidos. Los botones se emplearon para establecer la conexión con el microcontrolador a través de bluetooth al igual que para desconectarse; también se usaron para hacer el control manual de UGV y para iniciar cualquier modo de este. Los paneles se usaron para poder agrupar una cantidad de botones radiales, botones y texto estático de acuerdo con su función, estas funciones son el modo que va a tener el UGV, la velocidad de este, los botones de la conexión y desconexión y el ultimo que es el que te dice cuanta distancia esta moviendo el sensor. Finalmente, la GUI quedo de la siguiente manera:

Imagen que contiene Calendario

Descripción generada automáticamente

Cada uno de los botones tiene una función diferente, para comenzar primero damos clic en el botón conectar para iniciar nuestra comunicación, después le damos clic en el botón de PLAY para obtener los valores de 0/1 de los radiobutton dentro de la selección de modo.

Si selecciona el modo demostración mandara un dato al PIC para que comience con la rutina de movimiento automática.

Si selecciona el modo medición, mandara un dato para que este mida con el módulo ultrasónico y devuelva el valor medido por el programa del PIC al Matlab, lo obtenga y lo muestre en el statictext.

Si selecciona el modo Sig Lineas, mandara un dato para que el PIC comience la detección por medio de los sensores seguidores de línea de manera automática.

Si selecciona el modo control, mandara un dato para que el PIC comience un ciclo interno y así poder seleccionar la dirección y velocidad con Matlab. Dentro de este modo tendrá 5 botones para la operación de este, cada botón contiene un tipo de movimiento y adjunto a este la selección de la velocidad donde se elige por medio de los valores de 0/1 de los radiobutton dentro del modo control. Pose también un botón central que sirve de cancelación de ese mismo ciclo interno y así seleccionar otro modo.

Finalmente, para terminar todo el proceso se selecciona el botón desconectar para terminar con toda comunicación con el PIC.

Diseño

* Diagrama de flujo

Diagrama

Descripción generada automáticamente

* Diagrama eléctrico

Diagrama

Descripción generada automáticamente

* Esquemáticos
* Esquemático en el software Proteus

Diagrama, Esquemático

Descripción generada automáticamente

* Esquemático en el software Eagle

Diagrama, Esquemático

Descripción generada automáticamente

* Board del PCB en el software Eagle

Imagen que contiene circuito, reloj, alimentos

Descripción generada automáticamente

Código

* MPLAB HITECH

|  |
| --- |
| //LIBRERIAS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <htc.h>  #define \_LEGACY\_HEADERS  \_\_CONFIG (FOSC\_INTRC\_NOCLKOUT & WDTE\_OFF & PWRTE\_OFF & MCLRE\_ON& CP\_OFF & CPD\_OFF & BOREN\_OFF & IESO\_OFF & FCMEN\_OFF & LVP\_OFF & DEBUG\_OFF);  \_\_CONFIG (BOR4V\_BOR40V & WRT\_OFF);  //COM SETTINGS  #define BAUD 9600  #define \_XTAL\_FREQ 4000000  #define baudsetting ((int)(FOSC/(64UL \* BAUD) -1))  //PUERTOS USART  #define RX\_PIN TRISC7  #define TX\_PIN TRISC6  //PUERTOS HC-SR04  #define PIN\_ECHO RB0  #define PIN\_TRIG RB1  //PUERTOS LINE  #define PIN\_OUT1 RB7  #define PIN\_OUT2 RB6  //PUERTOS MOTOR  #define OUT1 RA0  #define OUT2 RA1  #define OUT3 RA2  #define OUT4 RA3  //TMR0 CONFIGURACION  void pause (unsigned short usvalue);  void msecbase(void);  //USART CONFIGURACION  void putch(unsigned char);  unsigned char getch(void);  void init\_comms(void);  //MOTOR/PWM/ADC CONFIGURACION  void PWM();  //ULTRASONIC CONFIGURACION  void SR();  //RUTINA CONFIGURACION  void rutina();  //LINE CONFIGURACION  void line();  //USART FUNCIONES  void putch(unsigned char byte){  /\* output one byte \*/  while(!TXIF) /\* set when register is empty \*/  continue;  TXREG = byte;  }  unsigned char getch(){  /\* retrieve one byte \*/  while(!RCIF) /\* set when register is not empty \*/  continue;  return RCREG;  }  void init\_comms(void){  RX\_PIN = 1;  TX\_PIN = 0;  SPBRG = 0x19;  //Continuous 8 bit asynchronous non inverted low speed communication  RCSTA = 0x90; // SPEN and CREN bit = 1, RX9EN = 0  TXSTA = 0x26;//TXEN = 1, BRGH, SYNC = 0  BAUDCTL = 0; //BRG16 = 0  }  //HC-SR04 VARIABLES  unsigned char carac1[4];  int dato;  unsigned int dist=0x00;  //MOTOR/PWM/ADC VARIABLES  unsigned int pwm1=0;  unsigned int pwm2=0;  //GENERAL VARIABLES  unsigned char op;  unsigned char vel;  unsigned char dir=1;  //INICIO CODIGO  void main(void){  ANSEL = 0x60;  ANSELH = 0x00;  TRISB = 0xFF;  TRISB0 = 1;  TRISB1 = 0;  TRISB7 = 1;  TRISB6 = 1;  TRISA = 0x00;  TRISD = 0x00;  TRISC = 0x00;  TRISE = 0x07;  ADCON1 = 0xFF;  //INICIO DE TMR1 PARA HC-SR04  T1CON = 0x00;  TMR1H=0x00;  TMR1L=0x00;  //INICIO DE TMR2 PARA MOTOR/PWM  T2CON = 0x07;  PR2 = 0xFF;  CCPR1L = 0x80;  CCP1CON = 0x0C;  ADFM = 0;  //LIMPIEZA PUERTOS  PIN\_TRIG=0;  PIN\_ECHO=0;  //INICIO COMUNICACION USART  init\_comms();  //INICIO CICLO PRINCIPAL  while(1){  //LIMPIEZA PUERTOS  OUT1 = 0;  OUT2 = 0;  OUT3 = 0;  OUT4 = 0;  TMR1H=0x00;  TMR1L=0x00;  dist=0;  dir=1;  op = getch();  //ELLECCION OPCIONES  switch(op){  case '1': //MEDICION DISTANCIA    SR();  break;  case '2': //RUTINA  rutina();  break;  case '3': //MOVIMIENTO BT  while(dir!='0'){  dir=getch();  vel=getch();    if(vel=='1'){  pwm1=500;  pwm2=500;  PWM();  }else if(vel=='2'){  pwm1=765;  pwm2=765;  PWM();  }else if(vel=='3'){  pwm1=1023;  pwm2=1023;  PWM();  }  if(dir=='1'){  OUT1 = 1;  OUT3 = 1;  \_\_delay\_ms(300);  OUT1 = 0;  OUT3 = 0;  }else if(dir=='2'){  OUT2 = 1;  OUT4 = 1;  \_\_delay\_ms(300);  OUT2 = 0;  OUT4 = 0;  }else if(dir=='3'){  OUT1 = 1;  OUT3 = 0;  \_\_delay\_ms(300);  OUT1 = 0;  OUT3 = 0;  }else if(dir=='4'){  OUT1 = 0;  OUT3 = 1;  \_\_delay\_ms(300);  OUT1 = 0;  OUT3 = 0;  }  }    break;  case '4': //SEGUIDOR DE LINEAS  for(int x=0;x<1000;x++){  line();  \_\_delay\_ms(50);  OUT1 = 0;  OUT2 = 0;  OUT3 = 0;  OUT4 = 0;  }  break;  default:  op = 0; //OPCION DEFAULT SI NO ES OPCION  break;  }  \_\_delay\_us(50);  }  }  //FUNCIONES LINEA  void line(){  if((PIN\_OUT1 == 0) && (PIN\_OUT2 == 0)){  pwm1=700;  pwm2=700;  PWM();  OUT1 = 1;  OUT3 = 1;  }else{  if((PIN\_OUT1 == 1) && (PIN\_OUT2 == 0)){  pwm1=700;  pwm2=100;  PWM();  OUT1 = 1;  OUT3 = 1;  }else{  if((PIN\_OUT1 == 0) && (PIN\_OUT2 == 1)){  pwm1=100;  pwm2=700;  PWM();  OUT1 = 1;  OUT3 = 1;  }else{  pwm1=700;  pwm2=700;  PWM();  OUT2 = 1;  OUT4 = 1;  }  }  }  }  //FUNCIONES ULTRASONICO  void SR(){  PIN\_TRIG=0;  \_\_delay\_us(2);  PIN\_TRIG=1;  \_\_delay\_us(10);  PIN\_TRIG=0;  while(!PIN\_ECHO);  TMR1ON=1;  while(PIN\_ECHO && !TMR1IF);  TMR1ON=0;  if(!TMR1IF){  dist = (TMR1H<<8 | TMR1L);  }else{  dist=0;  TMR1IF=0;  }    dist = (dist)/58;  if(dist<3 || 390<dist){  dato = 0;  printf("%d",dato);  }else if(3<dist || dist<390){  sprintf(carac1, "%d",dist);  dato = atoi(carac1);  printf(" %d ",dato);  }  }  //FUNCION RUTINA  void rutina(){  pwm1=700;  pwm2=700;  PWM();  OUT1 = 1;  OUT3 = 1;  \_\_delay\_ms(2000);  OUT1 = 0;  OUT3 = 0;  \_\_delay\_ms(500);  OUT2 = 1;  OUT4 = 1;  \_\_delay\_ms(2000);  OUT2 = 0;  OUT4 = 0;  \_\_delay\_ms(500);  OUT1 = 1;  OUT3 = 0;  \_\_delay\_ms(2000);  OUT1 = 0;  OUT3 = 0;  \_\_delay\_ms(500);  OUT2 = 1;  OUT4 = 0;  \_\_delay\_ms(2000);  OUT2 = 0;  OUT4 = 0;  \_\_delay\_ms(500);  OUT1 = 0;  OUT3 = 1;  \_\_delay\_ms(2000);  OUT1 = 0;  OUT3 = 0;  \_\_delay\_ms(500);  OUT2 = 0;  OUT4 = 1;  \_\_delay\_ms(2000);  OUT2 = 0;  OUT4 = 0;  \_\_delay\_ms(500);  }  //FUNCIONES MOTOR/ADC/PWM  void PWM(){ //PWM  ADRESH=pwm1>>2;  ADRESL=pwm1<<8;  CCPR1L=ADRESH;  CCP1CON=0X0C|(ADRESL>>2);  \_\_delay\_us(10);    ADRESH=pwm2>>2;  ADRESL=pwm2<<8;  CCPR2L=ADRESH;  CCP2CON=0X0C|(ADRESL>>2);  \_\_delay\_us(10);  }  void pause (unsigned short usvalue){  unsigned short x;  for (x=0; x<=usvalue; x++)  {  msecbase();  }  }  void msecbase(void){  OPTION\_REG = 0b11010010;  TMR0 = 0x0E;  TMR0 = 0x8E;  while(!T0IF);  T0IF = 0;  } |

* MATLAB

|  |
| --- |
| function varargout = BTCAR(varargin)  % BTCAR MATLAB code for BTCAR.fig  % BTCAR, by itself, creates a new BTCAR or raises the existing  % singleton\*.  %  % H = BTCAR returns the handle to a new BTCAR or the handle to  % the existing singleton\*.  %  % BTCAR('CALLBACK',hObject,eventData,handles,...) calls the local  % function named CALLBACK in BTCAR.M with the given input arguments.  %  % BTCAR('Property','Value',...) creates a new BTCAR or raises the  % existing singleton\*. Starting from the left, property value pairs are  % applied to the GUI before BTCAR\_OpeningFcn gets called. An  % unrecognized property name or invalid value makes property application  % stop. All inputs are passed to BTCAR\_OpeningFcn via varargin.  %  % \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one  % instance to run (singleton)".  %  % See also: GUIDE, GUIDATA, GUIHANDLES  % Edit the above text to modify the response to help BTCAR  % Last Modified by GUIDE v2.5 29-May-2022 00:10:15  % Begin initialization code - DO NOT EDIT  gui\_Singleton = 1;  gui\_State = struct('gui\_Name', mfilename, ...  'gui\_Singleton', gui\_Singleton, ...  'gui\_OpeningFcn', @BTCAR\_OpeningFcn, ...  'gui\_OutputFcn', @BTCAR\_OutputFcn, ...  'gui\_LayoutFcn', [] , ...  'gui\_Callback', []);  if nargin && ischar(varargin{1})  gui\_State.gui\_Callback = str2func(varargin{1});  end  if nargout  [varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});  else  gui\_mainfcn(gui\_State, varargin{:});  end  % End initialization code - DO NOT EDIT  % --- Executes just before BTCAR is made visible.  function BTCAR\_OpeningFcn(hObject, eventdata, handles, varargin)  % This function has no output args, see OutputFcn.  % hObject handle to figure  % eventdata reserved - to be defined in a future version of MATLAB  % handles structure with handles and user data (see GUIDATA)  % varargin command line arguments to BTCAR (see VARARGIN)  % Choose default command line output for BTCAR  handles.output = hObject;  axes(handles.axes2);  [x,map]=imread('fondo.jfif');  image(x)  colormap(map);  axis off  hold on  handles.output = hObject;  axes(handles.axes3);  [x,map]=imread('fondo1.png');  image(x)  colormap(map);  axis off  hold on  handles.output = hObject;  axes(handles.axes7);  [x,map]=imread('fondo1.png');  image(x)  colormap(map);  axis off  hold on  handles.output = hObject;  axes(handles.axes8);  [x,map]=imread('fondo1.png');  image(x)  colormap(map);  axis off  hold on  handles.output = hObject;  axes(handles.axes9);  [x,map]=imread('fondo1.png');  image(x)  colormap(map);  axis off  hold on  [a,map]=imread('derecha.jpg');  [r,c,d]=size(a);  x=ceil(r/60);  y=ceil(c/150);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.up,'CData',g)  [a,map]=imread('izquierda.jpg');  [r,c,d]=size(a);  x=ceil(r/60);  y=ceil(c/150);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.down,'CData',g)  [a,map]=imread('abajo.jpg');  [r,c,d]=size(a);  x=ceil(r/60);  y=ceil(c/92);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.der,'CData',g)  [a,map]=imread('arriba.jpg');  [r,c,d]=size(a);  x=ceil(r/60);  y=ceil(c/92);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.izq,'CData',g)  [a,map]=imread('play.jpg');  [r,c,d]=size(a);  x=ceil(r/75);  y=ceil(c/92);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.play,'CData',g)  [a,map]=imread('pausa.jpg');  [r,c,d]=size(a);  x=ceil(r/50);  y=ceil(c/60);  g=a(1:x:end,1:y:end,:);  g(g==255)=0.8\*255;  set(handles.cerrar,'CData',g)  % Update handles structure  guidata(hObject, handles);  % UIWAIT makes BTCAR wait for user response (see UIRESUME)  % uiwait(handles.figure1);  % --- Outputs from this function are returned to the command line.  function varargout = BTCAR\_OutputFcn(hObject, eventdata, handles)  % varargout cell array for returning output args (see VARARGOUT);  % hObject handle to figure  % eventdata reserved - to be defined in a future version of MATLAB  % handles structure with handles and user data (see GUIDATA)  % Get default command line output from handles structure  varargout{1} = handles.output;  % --- Executes on button press in up.  function up\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  L='1';  fprintf(SerialP,'%c',L);  x = get(handles.lento,'Value')  y = get(handles.regular,'Value')  z = get(handles.rapido,'Value')  if (x == 1) && (y == 0) && (z == 0)  L='1';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 1) && (z == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 0) && (z == 1)  L='3';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in izq.  function izq\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  L='4';  fprintf(SerialP,'%c',L);  x = get(handles.lento,'Value')  y = get(handles.regular,'Value')  z = get(handles.rapido,'Value')  if (x == 1) && (y == 0) && (z == 0)  L='1';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 1) && (z == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 0) && (z == 1)  L='3';  fprintf(SerialP,'%c',L);  else  L='1';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in down.  function down\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  L='2';  fprintf(SerialP,'%c',L);  x = get(handles.lento,'Value')  y = get(handles.regular,'Value')  z = get(handles.rapido,'Value')  if (x == 1) && (y == 0) && (z == 0)  L='1';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 1) && (z == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 0) && (z == 1)  L='3';  fprintf(SerialP,'%c',L);  else  L='1';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in der.  function der\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  L='3';  fprintf(SerialP,'%c',L);  x = get(handles.lento,'Value')  y = get(handles.regular,'Value')  z = get(handles.rapido,'Value')  if (x == 1) && (y == 0) && (z == 0)  L='1';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 1) && (z == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 0) && (z == 1)  L='3';  fprintf(SerialP,'%c',L);  else  L='1';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in cerrar.  function cerrar\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  L='0';  fprintf(SerialP,'%c',L);  x = get(handles.lento,'Value')  y = get(handles.regular,'Value')  z = get(handles.rapido,'Value')  if (x == 1) && (y == 0) && (z == 0)  L='1';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 1) && (z == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (x == 0) && (y == 0) && (z == 1)  L='3';  fprintf(SerialP,'%c',L);  else  L='1';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in play.  function play\_Callback(hObject, eventdata, handles)  global SerialP;  fopen(SerialP);  a = get(handles.demostracion,'Value')  b = get(handles.medicion,'Value')  c = get(handles.sg,'Value')  d = get(handles.control,'Value')  if (a == 1) && (b == 0) && (c == 0) && (d == 0)  L='2';  fprintf(SerialP,'%c',L);  elseif (a == 0) && (b == 1) && (c == 0) && (d == 0)  L='1';  fprintf(SerialP,'%c',L);  dato = fscanf(SerialP,'%d');  set(handles.dist,'String',num2str(dato));  elseif (a == 0) && (b == 0) && (c == 1) && (d == 0)  L='4';  fprintf(SerialP,'%c',L);  elseif (a == 0) && (b == 0) && (c == 0) && (d == 1)  L='3';  fprintf(SerialP,'%c',L);  else  L='0';  fprintf(SerialP,'%c',L);  end  fclose(SerialP);  % --- Executes on button press in lento.  function lento\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of lento  % --- Executes on button press in regular.  function regular\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of regular  % --- Executes on button press in rapido.  function rapido\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of rapido  % --- Executes on button press in demostracion.  function demostracion\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of demostracion  % --- Executes on button press in medicion.  function medicion\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of medicion  % --- Executes on button press in sg.  function sg\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of sg  % --- Executes on button press in control.  function control\_Callback(hObject, eventdata, handles)  % Hint: get(hObject,'Value') returns toggle state of control  % --- Executes on button press in conectar.  function conectar\_Callback(hObject, eventdata, handles)  global SerialP;  SerialP = serialport("COM3",9600,"StopBits",1,"DataBits",8,"Parity","none","FlowControl","none","Timeout",1);  % --- Executes on button press in desconectar.  function desconectar\_Callback(hObject, eventdata, handles)  global SerialP;  fclose(SerialP);  delete(SerialP);  % --- Executes on button press in pausa.  function pausa\_Callback(hObject, eventdata, handles)  % hObject handle to pausa (see GCBO)  % eventdata reserved - to be defined in a future version of MATLAB  % handles structure with handles and user data (see GUIDATA) |

Conclusión

Finalmente, como se habló al principio de este trabajo los microprocesadores y microcontroladores forman parte de los dispositivos electrónicos más importantes que hay en la actualidad debido a que tienen una amplia aplicación en muchos artículos que usamos día con día y que como ingenieros saber cómo se comportan, sus características, su funcionamiento y sus aplicaciones en la tecnología actual nos puede ser de mucha utilidad por lo que al desarrollar este proyecto de un prototipo de UGV pudimos reafirmar y terminar de comprender los temas que se vieron en la unidad de aprendizaje microprocesadores, microcontroladores e interfaz. Durante el desarrollo del proyecto se tuvieron algunos contratiempos al momento de la programación como lo fue con el sensor ultrasónico al momento de que se queríamos mostrar la distancia medida en la hiperterminal nos mostraba símbolos que no tenían nada que ver con lo que deseábamos debido a que no estaba siendo traducido el dato por el código y también con los seguidores de línea se tuvieron algunos problemas al momento de la calibración, pero esto no fue gran problema.

Adicional a esto, también se aplicaron conocimientos de unidades de aprendizaje anteriores como la de desarrollo de circuitos impresos con lo que se obtuvo una reafirmación de esos conocimientos al ser implementados en un proyecto más completo, así puliendo la habilidad del diseño y la transferencia de los circuitos impresos en las placas fenólicas.

Anexos

* Fotografías del vehículo terrestre no tripulado

Imagen que contiene pequeño, tabla, naranja, azul

Descripción generada automáticamente Imagen que contiene interior, tabla, pequeño, escritorio

Descripción generada automáticamente

Imagen que contiene interior, tabla, juguete, pequeño

Descripción generada automáticamente Juguete de plástico de colores

Descripción generada automáticamente con confianza baja

Imagen que contiene interior, tabla, juguete, pequeño

Descripción generada automáticamente

* Fotografías de la placa de circuito impreso

Un letrero de color negro

Descripción generada automáticamente con confianza media

Fuentes

[1] O. C. Vargas, N. Farías, E. Flores, C. E. Maciel y E. Cárdenas. "Tecnologías bluetooth aplicadas al control de vehículos terrestres no tripulados para aspersión de agroquímicos en plantas ornamentales". Sistema de Información Científica Redalyc, Red de Revistas Científicas. https://www.redalyc.org/journal/614/61452676006/html/#redalyc\_61452676006\_r (accedido el 31 de mayo de 2022).

[2] S. Sánchez. "Los microcontroladores de hoy en día". Microcontroladores. https://microcontroladoressesv.wordpress.com/los-microcontroladores-de-hoy-en-dia/ (accedido el 31 de mayo de 2022).

[3] J. E. Joffré. "Guia para la redaccion de reportes tecnicos". Facultad de Ingeniería Facultad de Ingeniería. http://www.ingenieria.uaslp.mx/Metalurgia/Documents/APUNTES/Guía%20para%20escribir%20reporte.pdf (accedido el 31 de mayo de 2022).