UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Car with Bluetooth communication

Jesús Baltasar Fernández
Jaime Fernández-Bravo Carranza
Carlos Gómez Fernández

Asignatura: Diseño de Sistemas Basados en Microprocesadores

Titulación: Grado en Ingeniería Informática

Fecha: 6 de Junio de 2021

ÍNDICE

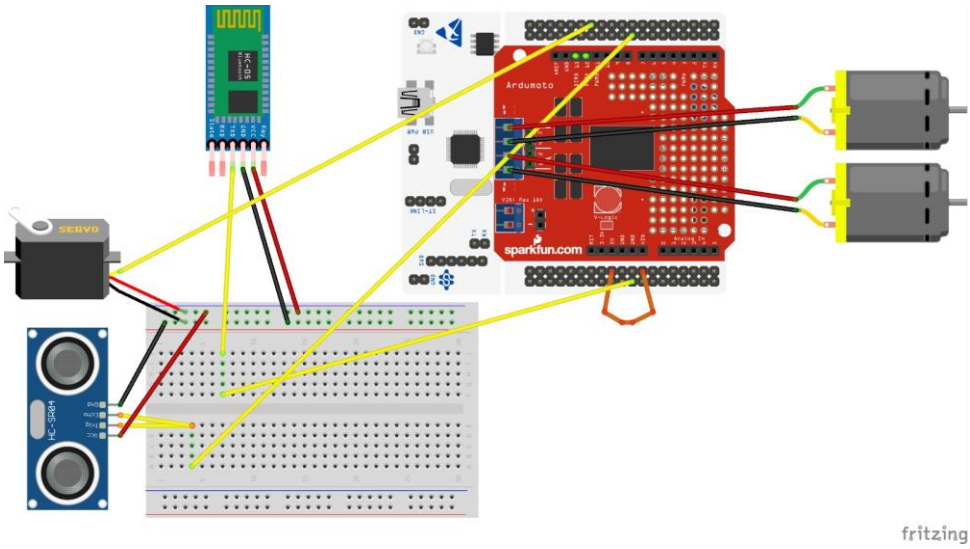
1. MATERIALES.

Para la realización del proyecto se ha empleado el siguiente hardware:

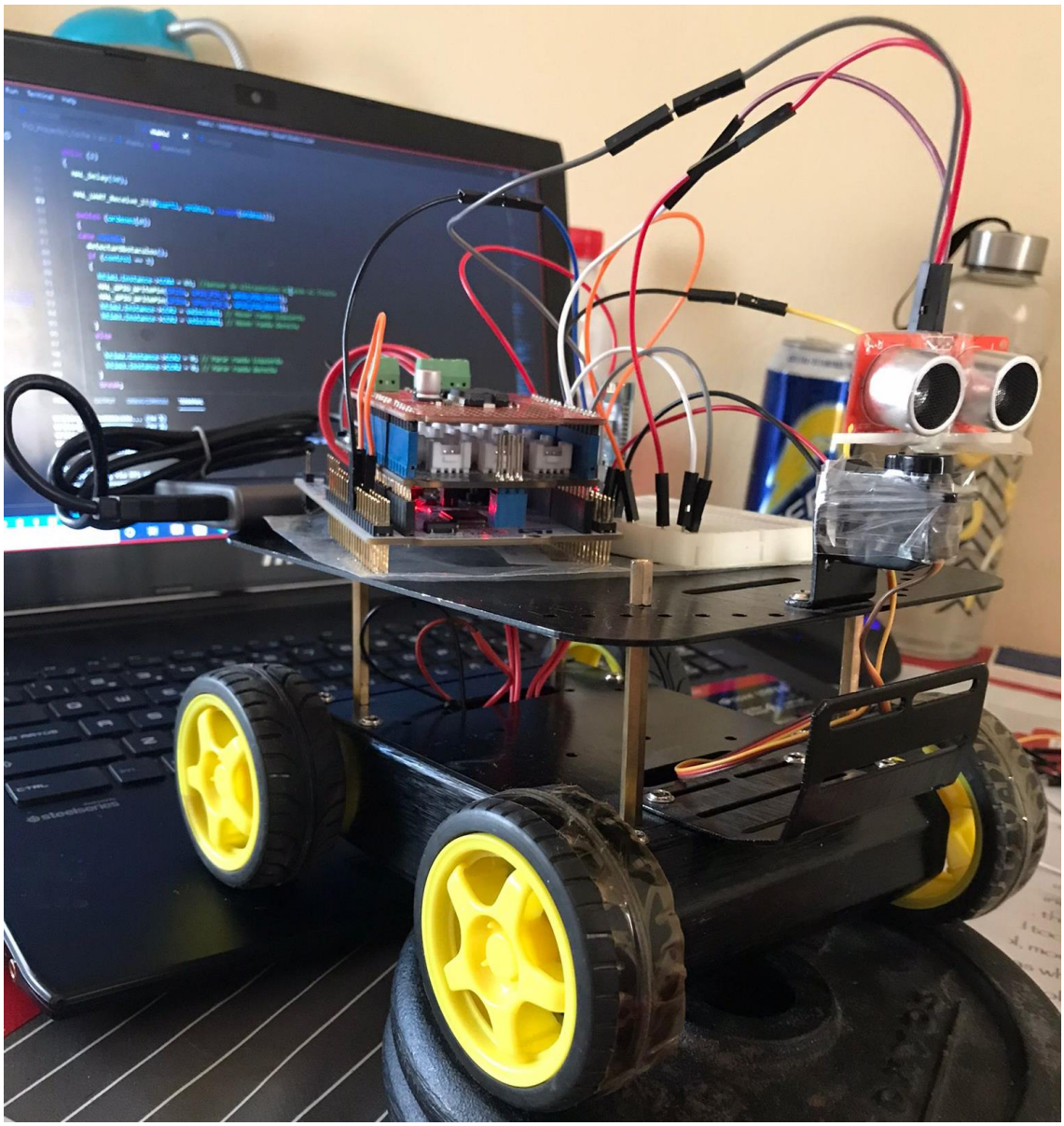
- STM32F-411RE
- Placa de expansión Grove
- Placa Ardumoto
- Módulo bluetooth HC-05
- Servo-motor
- Sensor de ultrasonidos
- Base con dos ruedas motrices accionadas con dos servo-motores.
- Powerbank
- Conector USB A-Mini USB
- Cables Dupont (x14)
- Protoboard

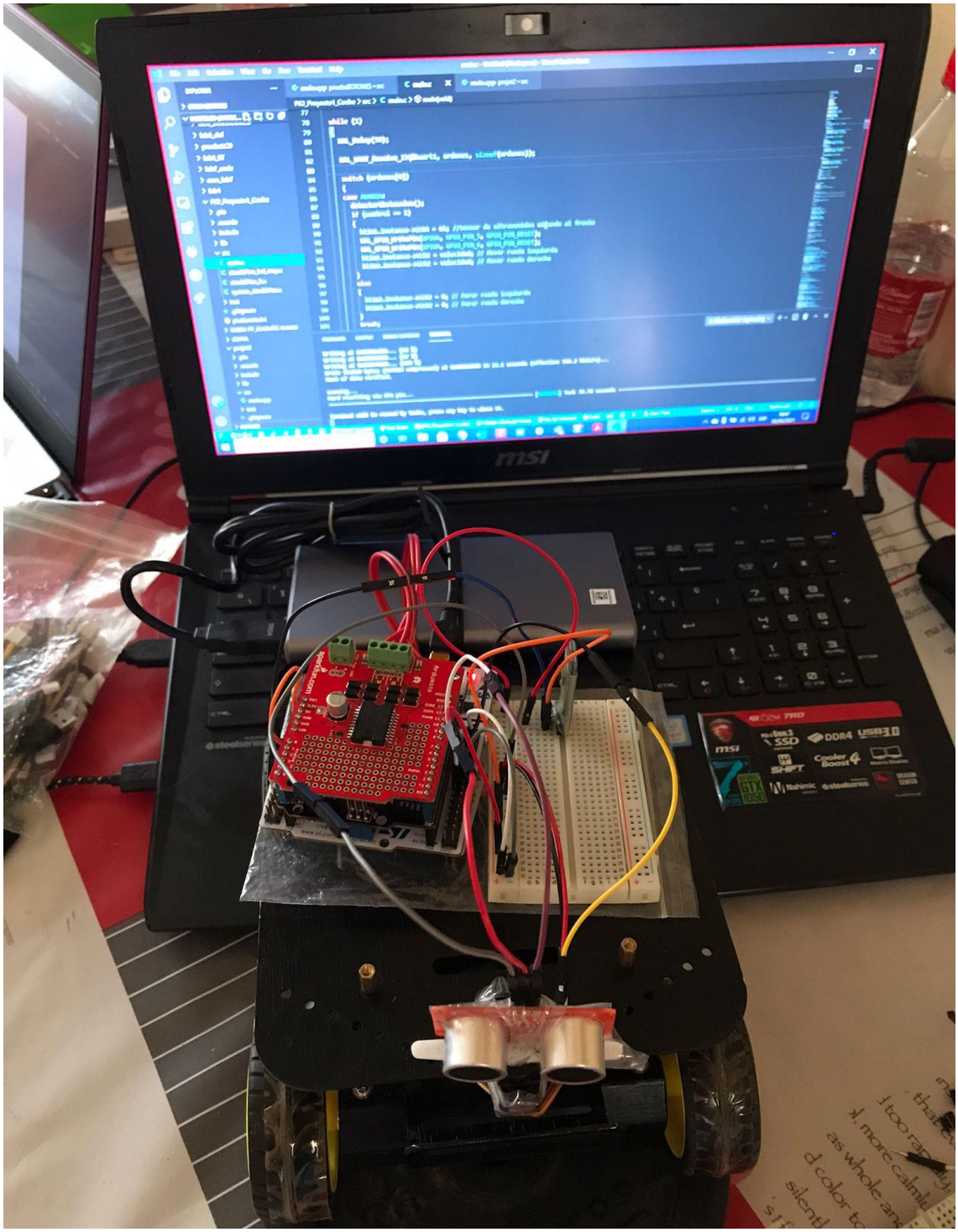
2. DISEÑO.

2.1. ESQUEMA DE DISEÑO.

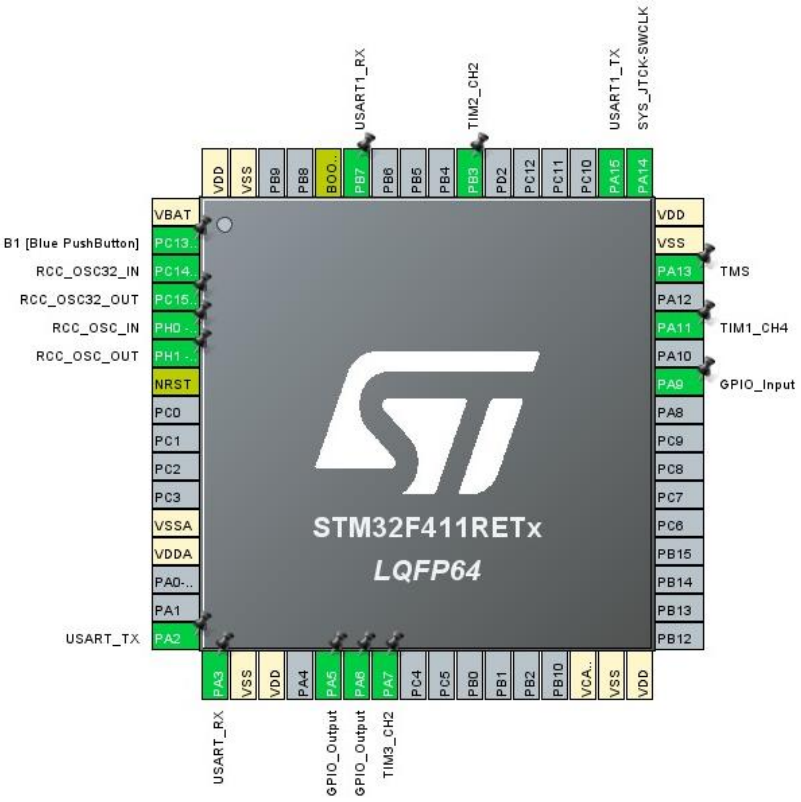


2.2. DISEÑO HARDWARE.





3. CONFIGURACIÓN DE LOS PINES EM STM32CUBEMX.



PIN	ESTADO	FUNCIONALIDAD
PA9	GPIO_INPUT	Echo y trigger del sensor de ultrasonidos
PA5	GPIO_OUTPUT	Activar el movimiento del servo de la rueda derecha
PA6	GPIO_OUTPUT	Activar el movimiento del servo de la rueda izquierda

PIN	ESTADO	CHANNEL	PERIODO	PRESCALER	FUNCIONALIDAD
PA11	TIM1_CH4	PWM generation CH4	999	1679	Mover el servo que controla la dirección del sensor de ultrasonido
PB3	TIM2_CH2	PWM generation CH2	999	1679	Mover el servo que controla el movimiento de la rueda izquierda
PA7	TIM3_CH2	PWM generation CH2	999	1679	Mover el servo que controla el movimiento de la rueda derecha

PIN	SEÑAL	GPIO_OUTPUT LEVEL	GPIO MODE	GPIO PULL-UP/PULL-DOWN	MAXIMUM OUTPUT SPEED	USER LABEL	MODIFIED
PA2	USART2_TX	n/a	Alternative function push pull	No pull-up and no pull-down	Very high	USART_TX	Yes
PA3	USART2_RX	n/a	Alternative function push pull	No pull-up and no pull-down	Very high	USART_RX	Yes
PA15	USART1_TX	n/a	Alternative function push pull	No pull-up and no pull-down	Very high	USART_TX	Yes
PB7	USART1_RX	n/a	Alternative function push pull	No pull-up and no pull-down	Very high	USART_TX	Yes

4. FUNCIONALIDAD.

La funcionalidad básica de este proyecto es conseguir controlar el coche mediante *bluetooth*, haciendo uso de una aplicación móvil, en nuestro caso, *BlueDuino*, y el módulo *bluetooth* mencionado anteriormente en la sección de materiales. Además de la dirección, otro aspecto a controlar es la velocidad del coche, la cual será regulada desde la aplicación, pudiendo tomar hasta tres valores distintos.

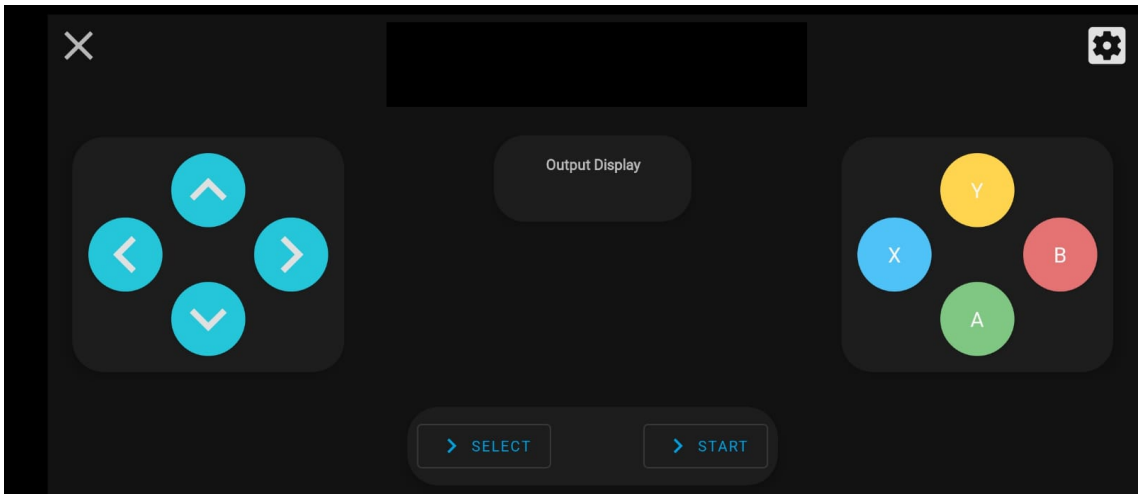


Figura 1. Mando *bluetooth* de la aplicación *BlueDuino*

Para llevar a cabo la comunicación *bluetooth* entre el coche y la aplicación, esta enviará caracteres que indicarán tanto la velocidad como la dirección del coche, los cuales se detallan a continuación:

- Para que el coche avance, la aplicación enviará el carácter "w"
- Para que el coche retroceda, la aplicación enviará el carácter "s"
- Para que el coche gire hacia la izquierda, la aplicación enviará el carácter "a"
- Para que el coche gire hacia la derecha, la aplicación enviará el carácter "d"
- Para que el coche se mueva con la velocidad más baja, la aplicación enviará el carácter "z"
- Para que el coche se mueva con la velocidad intermedia, la aplicación enviará el carácter "x"
- Para que el coche se mueva con la velocidad más alta, la aplicación enviará el carácter "c"
- Para que el coche no se mueva, la aplicación enviará el carácter "0"

Todas las ordenes recibidas por el microcontrolador mediante *bluetooth*, se recibirán mediante interrupciones.

Para conseguir esta funcionalidad, hemos hecho de una máquina de estados, la cual pasaremos a detallar a continuación. Esta máquina de estados dispone de un total de 8 estados, los cuales se corresponden con cada una de las acciones permitidas y comentadas anteriormente en la comunicación *bluetooth* entre aplicación y modulo *bluetooth*.

En todos los estados, menos en el cual el coche podrá retroceder, ya que, el coche no dispone de detector de obstáculos en la parte trasera, tendremos una variable entera "*control*", la cual podrá tomar valores de 0 y 1, la cual determinará si se puede o no mover el coche, en función de si se detectó un obstáculo o no.

El primer estado es en el cual el coche podrá avanzar ("*AVANZA*"), en él, el sensor de ultrasonidos mirará al frente y los pines a los que están conectados los servos estará ambos en *reset*, manteniendo la velocidad seleccionada en los estados de la máquina de estados habilitados para ello.

El segundo estado es en el cual el coche podrá retroceder ("*RETROCEDE*"), en él, el sensor de ultrasonidos mirará al frente y los pines a los que están conectados los servos estará ambos en *set*, manteniendo la velocidad seleccionada en los estados de la máquina de estados habilitados para ello.

El tercer estado es en el cual el coche podrá girar hacia la izquierda ("*IZQUIERDA*"), el pin que está conectado al servo de la rueda izquierda estará en *set*, en cambio el pin que está conectado al servo de la rueda derecha estará en *reset*, manteniendo la velocidad seleccionada en los estados de la máquina de estados habilitados para ello.

El cuarto estado es en el cual el coche podrá girar hacia la derecha ("*DERECHA*"), el pin que está conectado al servo de la rueda izquierda estará en *reset*, en cambio el pin que está conectado al servo de la rueda derecha estará en *set*, manteniendo la velocidad seleccionada en los estados de la máquina de estados habilitados para ello.

El quinto estado es en el cual se seleccionará la menor velocidad del coche ("*PRIMERA*"), se asignará a la variable velocidad el valor 500, el cual será el valor asociado al *timer*.

El sexto estado es en el cual se seleccionará la velocidad intermedia del coche ("*SEGUNDA*"), se asignará a la variable velocidad el valor 700, el cual será el valor asociado al *timer*.

El séptimo estado es en el cual se seleccionará la máxima velocidad del coche ("*TERCERA*"), se asignará a la variable velocidad el valor 900, el cual será el valor asociado al *timer*.

El octavo estado es en el cual el coche no se moverá ("*default*"), en él las velocidades tendrán el valor 0 consiguiendo así que no se mueva el coche.

Además del movimiento del coche vía *bluetooth*, era necesario implementar un detector de obstáculos, para ello se hemos optado por la solución que implicaba el uso de un sensor de ultrasonido y un servo-motor, cuya dirección siempre será la del movimiento del vehículo, para que al detectar un obstáculo a menos de 25 cm, el vehículo se pare, impidiendo que siga en dicha dirección, y sólo permitiendo al vehículo retroceder.

5. CÓDIGO.

5.1. MÁQUINA DE ESTADOS PARA CONTROLAR EL MOVIMIENTO DEL VEHÍCULO Y DEL SERVO MOTOR QUE POSICIONA EL SENSOR DE ULTRASONIDOS.

```
78  while (1)
79  {
80    HAL_Delay(50);
81
82    HAL_UART_Receive_IT(&huart1, ordenes, sizeof(ordenes));
83
84    switch (ordenes[0])
85    {
86    case AVANZA:
87      detectarObstaculos();
88      if (control == 1)
89      {
90        htim1.Instance->CCR4 = 65; //Sensor de ultrasonidos mirando al frente
91        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
92        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
93        htim2.Instance->CCR2 = velocidad; // Mover rueda izquierda
94        htim3.Instance->CCR2 = velocidad; // Mover rueda derecha
95      }
96      else
97      {
98        htim2.Instance->CCR2 = 0; // Parar rueda izquierda
99        htim3.Instance->CCR2 = 0; // Parar rueda derecha
100     }
101     break;
102   case RETROCEDE:
103     control = 1;
104     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
105     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
106     htim2.Instance->CCR2 = velocidad; // Mover rueda izquierda
107     htim3.Instance->CCR2 = velocidad; // Mover rueda derecha
108     break;
109   case IZQUIERDA:
110     detectarObstaculos();
111       if (control == 1)
112       {
113         htim1.Instance->CCR4 = 105; // Sensor de ultrasonidos mirando a la izquierda
114         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
115         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_SET);
116         htim2.Instance->CCR2 = velocidad; // Mover rueda izquierda
117         htim3.Instance->CCR2 = velocidad; // Mover rueda derecha
118       }
119       else
120       {
121         htim2.Instance->CCR2 = 0; // Parar rueda izquierda
122         htim3.Instance->CCR2 = 0; // Parar rueda derecha
123       }
124     break;
125   case DERECHA:
126     detectarObstaculos();
127       if (control == 1)
128       {
129         htim1.Instance->CCR4 = 35; // Sensor de ultrasonidos mirando a la derecha
```

```
130             HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
131             HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, GPIO_PIN_RESET);
132             htim2.Instance->CCR2 = velocidad; // Mover rueda izquierda
133             htim3.Instance->CCR2 = velocidad; // Mover rueda derecha
134           }
135           else
136           {
137             htim2.Instance->CCR2 = 0; // Parar rueda izquierda
138             htim3.Instance->CCR2 = 0; // Parar rueda derecha
139           }
140         break;
141       case PRIMERA:
142         velocidad = 500; // Primera marcha
143         break;
144       case SEGUNDA:
145         velocidad = 750; // Segunda marcha
146         break;
147       case TERCERA:
148         velocidad = 1000; // Tercera marcha
149         break;
150       default:
151         htim2.Instance->CCR2 = 0; // Mover rueda izquierda
152         htim3.Instance->CCR2 = 0; // Mover rueda derecha
153         break;
154       }
155     }
156   }
```

5.2. CONTROL DEL SENSOR DE ULTRASONIDOS.

5.2.1. FUNCIÓN PARA DETECTAR OBSTACULOS CERCANOS.

```
158 void detectarObstaculos()
159 {
160   int instante2 = 0;
161   int instante3 = 0;
162   cambiarModoPin(0); /*Modo output*/
163   instanteIni = 0;
164   HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, GPIO_PIN_SET);
165
166   while (instanteIni >= 1)
167   {
168   }
169
170   cambiarModoPin(1); /*Modo Input*/
171
172   while (!(GPIOA->IDR & GPIO_IDR_ID9_Msk))  // Esperamos mientras el Pin 9 tenga valor 0
173   {
174   }
175   instante2 = instanteIni;
176   while ((GPIOA->IDR & GPIO_IDR_ID9_Msk))   // Esperamos mientras el Pin 9 tenga valor 1
177   {
178   }
179   instante3 = instanteIni;
180
181   if (calcularDistancia(instante3 - instante2) < 25)
182   {
183     control = 0;
184   }
185   else
186   {
187     control = 1;
188   }
189 }
```

5.2.2.FUNCIÓN PARA MODIFICAR EL MODO DE TRABAJO DEL SENSOR DE ULTRASONIDOS.

```
191  void cambiarModoPin(int modo)
192  {
193    GPIO_InitTypeDef GPIO_InitStruct = {0};
194    if (modo == 0) // Modo output
195    {
196      GPIO_InitStruct.Pin = GPIO_PIN_9;
197      GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
198      GPIO_InitStruct.Pull = GPIO_NOPULL;
199      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
200    }
201    else if (modo == 1) //Modo input
202    {
203      GPIO_InitStruct.Pin = GPIO_PIN_9;
204      GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
205      GPIO_InitStruct.Pull = GPIO_NOPULL;
206      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
207    }
208  }
```

5.2.3.FUNCIÓN PARA CALCULAR LA DISTANCIA ENTRE EL OBSTÁCULO Y EL VEHÍCULO.

```
210  int calcularDistancia(int mediciones)
211  {
212    int distancia = 0;
213    distancia = (mediciones * 10) / 58;
214    HAL_Delay(10);
215    printf("distancia: %d cm\n",distancia);
216    return distancia;
217  }
```