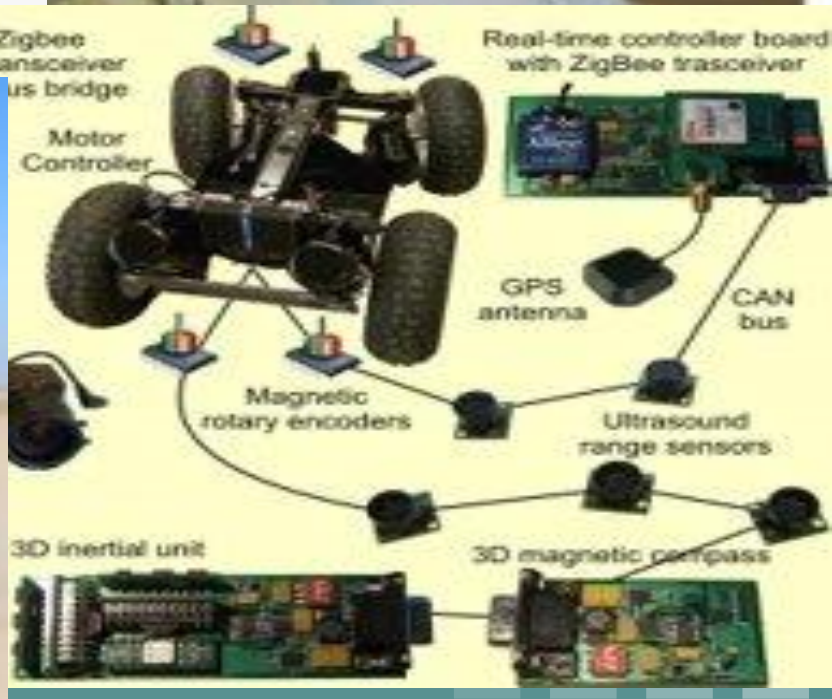
Curso de Especialización en Sistemas Embebidos

Sistemas Operativos en Tiempo Real

ALUMNO: Ing. Juan Pablo Menditto

DOCENTES Ing. Esp. Francisco BUCAFUSCO
Ing. Esp. Sergio De JESUS MELEÁN

Unmanned Ground Vehicle (UGV) Market Increasing the Growth Worldwide: Industry Analysis, Growth, Drivers, Limitations, Regions, Forecast 2022



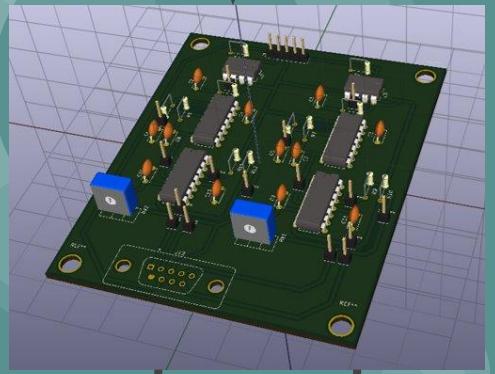
Nuevas Tecnologías

- Aplicación
- Funcionamiento
- Conclusiones

Aplicación

Control de velocidad

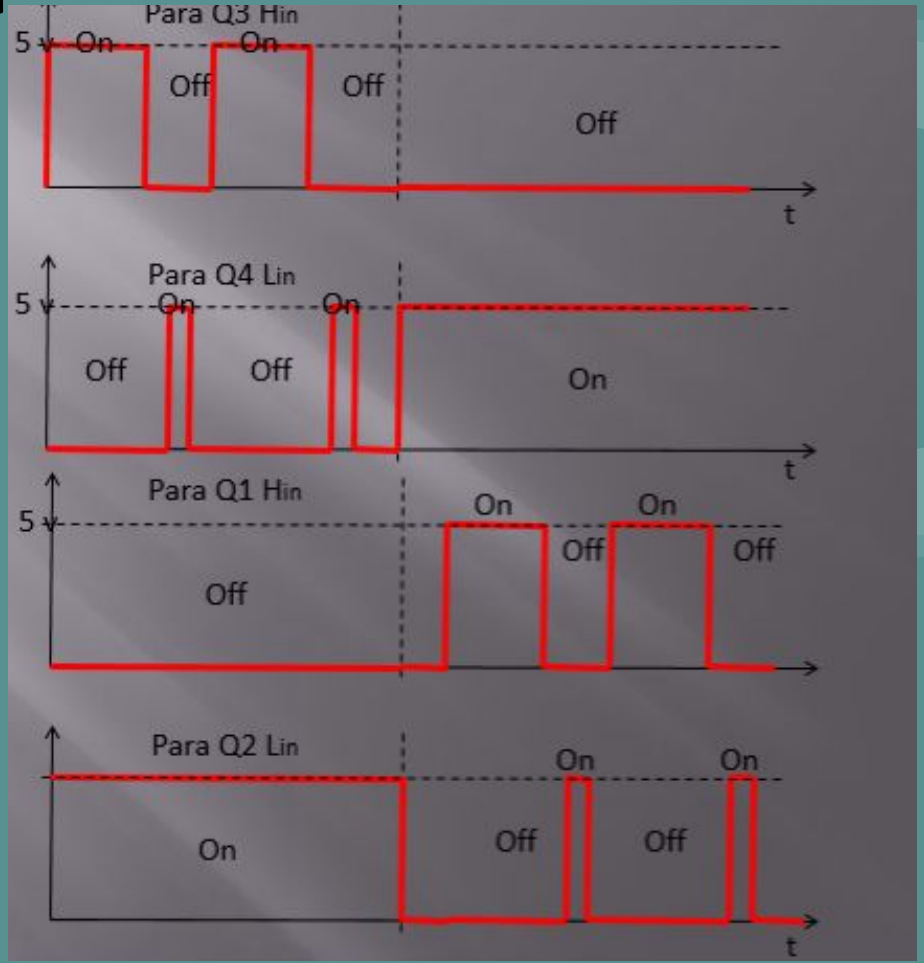
PWM



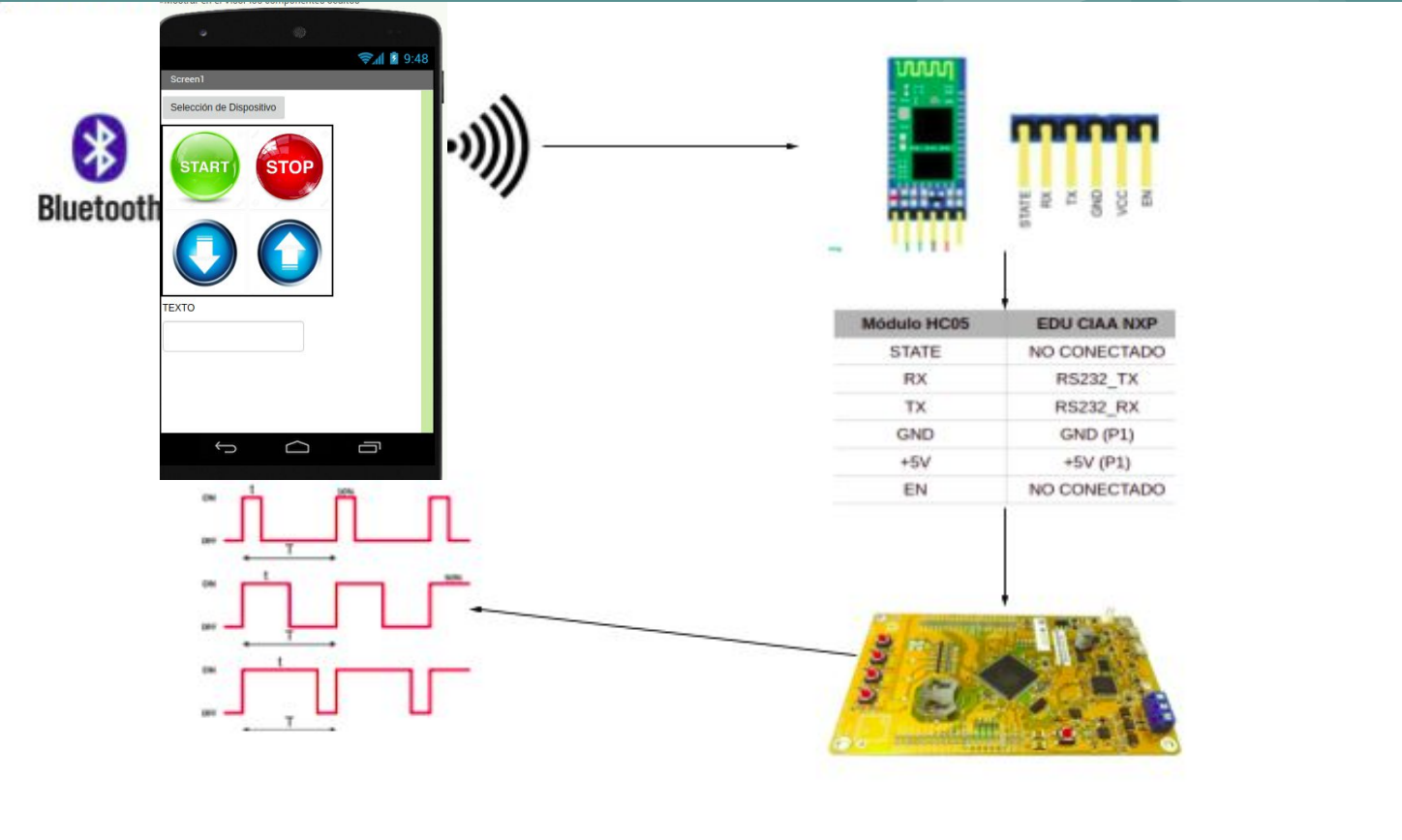
DRIVER

MOTOR

Aplicación


Para Q3 Hin
5 v
On
On
Off
Off
Off
t
Para Q4 Lin
5 v
On
On
Off
Off
On
t
Para Q1 Hin
5 v
On
On
Off
Off
Off
Para Q2 Lin
On
On
On
Off
Off
t

FUNCIONAMIENTO



Módulo HC05	EDU CIAA NXP
STATE	NO CONECTADO
RX	RS232_TX
TX	RS232_RX
GND	GND (P1)
+5V	+5V (P1)
EN	NO CONECTADO

FUNCIONAMIENTO

FUNCIONAMIENTO

Objetivo del RTOS

- ADMINISTRAR LAS TAREAS DE BT Y PULSOS
- Enlazar el BT mantener la conexión
- Recibir los datos de la APP de control
- HABILITAR PWM (Encendido/Apagado)
- Variar PWM
- Parada de emergencia, en caso de perder conexion.

FUNCIONAMIENTO

Implementación de Semaforo MUTEX

- ADMINISTRAR LAS TAREAS DE BT Y PULSOS
- Enlazar el BT mantener la conexión
- Recibir los datos de la APP de control
- HABILITAR PWM (Encendido/Apagado)
- Variar PWM
- Parada de emergencia, en caso de perder conexion.

FUNCIONAMIENTO

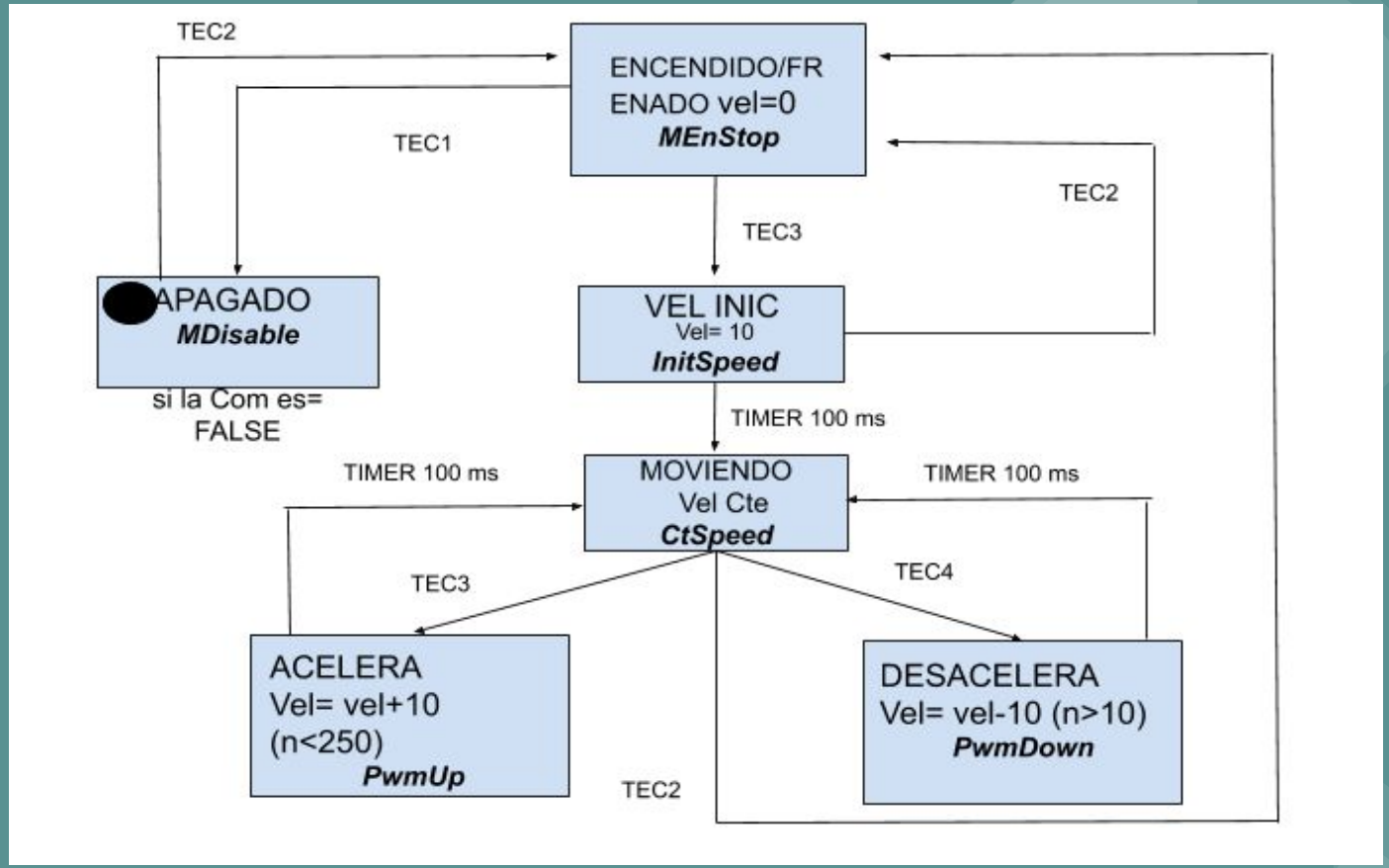
void ReceiveBT (void* taskParmPtr);

void pwmMotor(void* taskParmPtr);

VARIABLE = CONTROL

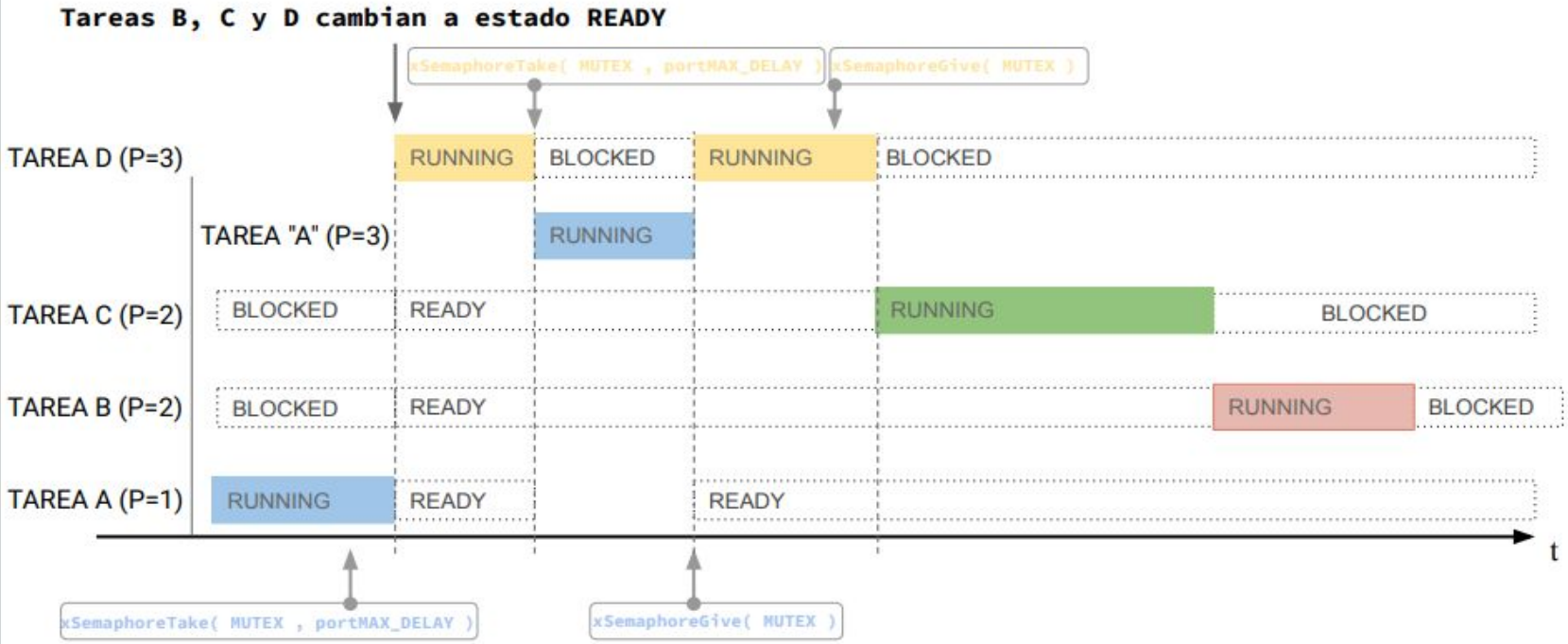
FUNCIONAMIENTO

void pwmMotor(void* taskParmPtr);



➢ FUNCIONAMIENTO

Resolución con Mutex



```
26 /*==================[definiciones de datos internos]==========================*/
27 SemaphoreHandle_t Evento_pulsado, Mutex_t_pulsado, Mutex_UART;
28 portTickType TiempoPulsado;
29 /*==================[definiciones de datos externos]==========================*/
30
31 uint8_t control = 0; //variable que indica la instruccion en el switch
32 DEBUG_PRINT_ENABLE;
33
34
35 /*==================[declaraciones de funciones internas]====================*/
36
37 /*==================[declaraciones de funciones externas]====================*/
38
39 void ReceiveBT ( void* taskParmPtr );
40
41 void pwmMotor( void* taskParmPtr );
42
43 /*==================[funcion principal]======================================*/
44
45 // FUNCION PRINCIPAL, PUNTO DE ENTRADA AL PROGRAMA LUEGO DE ENCENDIDO O RESET.
46 int main(void)
47 {
48    uint8_t Error_state = 0;  //variable local para registrar errores
49     // ---------- CONFIGURACIONES ------------------------------
50    // Inicializar y configurar la plataforma
51    boardConfig();
52
53    // Inicializar UART_USB para conectar a la PC
54    uartConfig( UART_PC, 9600 );
55    uartWriteString( UART_PC, "UART_PC configurada.\r\n" );
56
```

➢ FUNCIONAMIENTO

FUNCIONAMIENTO

```
81
82    xTaskCreate(
83      ReceiveBT,                      // Funcion de la tarea a ejecutar
84      (const char *)"ReceiveBT",      // Nombre de la tarea como String amigable para el usuario
85      configMINIMAL_STACK_SIZE*4, // Cantidad de stack de la tarea
86      0,                          // Parametros de tarea
87      tskIDLE_PRIORITY+1,         // Prioridad de la tarea
88      0                           // Puntero a la tarea creada en el sistema
89    );
90
91    // Crear tarea pwmMotor en freeRTOS
92
93    xTaskCreate(
94      pwmMotor,                       // Funcion de la tarea a ejecutar
95      (const char *)"pwmMotor",       // Nombre de la tarea como String amigable para el usuario
96      configMINIMAL_STACK_SIZE*2, // Cantidad de stack de la tarea
97      0,                          // Parametros de tarea
98      tskIDLE_PRIORITY+1,         // Prioridad de la tarea
99      0                           // Puntero a la tarea creada en el sistema
00    );
01
02
03
```

FUNCIONAMIENTO

```
if( uartReadByte( UART_BLUETOOTH, &data ) )
  {
   if( data == 'C' )
   {
       /*
      gpioWrite( LED1, ON );
      gpioWrite( LED2, OFF );
      gpioWrite( LED3, OFF );
      gpioWrite( LEDR, OFF );*/

      xSemaphoreTake(Mutex_UART,portMAX_DELAY);
      control = 1;
      xSemaphoreGive(Mutex_UART);
   }

   if( data == 'A' )
   {
       /*
       gpioWrite( LED1, OFF );
       gpioWrite( LED2, ON );
       gpioWrite( LED3, OFF );
       gpioWrite( LEDR, OFF );*/
       xSemaphoreTake(Mutex_UART,portMAX_DELAY);
       control = 2;
       xSemaphoreGive(Mutex_UART);
   }
   if( data == 'B' )
   {
       /*
```

```
    gpioWrite( LED2, FALSE);
    test2 = pwmConfig( PWM6, PWM_ENABLE_OUTPUT );
    pwmWrite( PWM6, dutyCycleInit );
    vTaskDelayUntil( &xLastWakeTime, xPeriodicity );
    if (test2)
        {
        gpioWrite( LED3, TRUE);
        gpioWrite( LED1, FALSE);
        }
    fT1 = FALSE;
    fT2 = TRUE;
    control = 0;

break;

case 3:

  if(fT1 == FALSE)  //es para incrementar pwm, verifica que se haya apretado la TEC2
    {
      if(dutyCycle0 <= 0)
      {
         dutyCycle0 = 5;              //es el valor inicial luego de que pwm sea cero
         vTaskDelayUntil( &xLastWakeTime, xPeriodicity );
      }
      else
       {
           if(dutyCycle0 < 250)
            {
              dutyCycle0 = dutyCycle0 + 10;
              pwmWrite( PWM6, dutyCycle0 );
              vTaskDelayUntil( &xLastWakeTime, xPeriodicity );
             }
```

DEMOSTRACION

PREGUNTAS

GRACIAS