

Diseño de sistemas en chip

SISTEMA DE INFOENTRETENIMIENTO

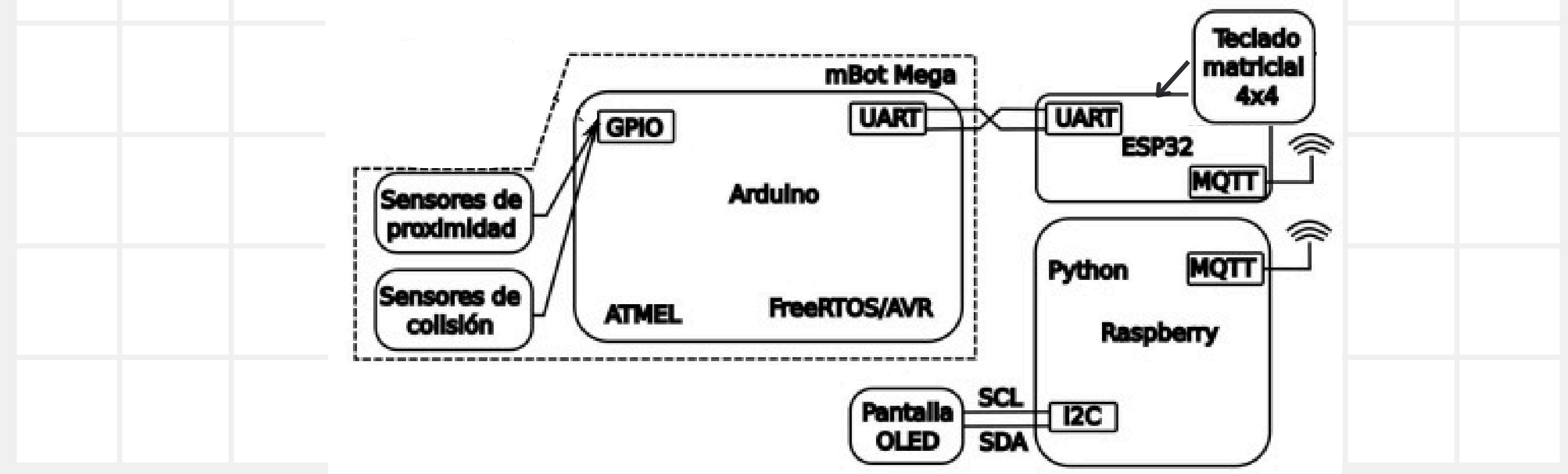
Oscar Ortiz Torres
A01769292

Yonathan
Romero Amador
A01737244



OBJETIVO

Desarrollar un sistema de control de vehículo (mBot Mega), cuyo microcontrolador debe estar programado en FreeRTOS, y un sistema de reproducción de audio (Raspberry), el cual recibirá y enviará información de control. El manejo de datos entre ambos sistemas será a través del ESP32.



USO DE FREERTOS EN ATMEGA2560

```
9 //Librerias y definiciones necesarias
10 #include <Arduino_FreeRTOS.h>
11 #include "queue.h"
12 #include <avr/io.h>
13 #include <util/delay.h>
14
15 #define F_CPU 16000000
16 #define USART_BAUDRATE 19200
17 #define UBRR_VALUE (((F_CPU / (USART_BAUDRATE * 16UL)) - 1)
18
19 //Declaración de variables
20 char RX_Byte;
21 unsigned char mybuffer[256];
22
23 //handle para un queue
24 QueueHandle_t UARTqueue;
25
26
27
28
29
30
31
32
33
34
35
36 void setup() {
37     //Creación de tareas
38     xTaskCreate(vUSARTTask, "USART MANAGER TASK", 100, NULL, 2, NULL);
39     xTaskCreate(vMBotSensorsTask, "MBot SENSORS TASK", 100, NULL, 1, NULL);
40
41     //creación de la queue
42     UARTqueue = xQueueCreate(10, sizeof(char));
43     if(UARTqueue != NULL) {
44         //configuración UART
45         UBRR2H = (uint8_t)(UBRR_VALUE >> 8);
46         UBRR2L = (uint8_t)UBRR_VALUE;
47         //UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01); // 8-bit data
48         UCSR2C = 0x06;           // Set frame format: 8data, 1stop bit
49         UCSR2B |= (1 << RXEN0) | (1 << TXEN0) | (1 << RXCIE0); // TX and RX enables and RX interrupt enabled
50         UCSR2A = 0x00; // Limpia banderas
51         sei();
52     }
```

CONTROL DEL REPRODUCTOR CON TECLADO MATRICIAL

ESP32

```
115 //Control de la variable repFlag con la tecla de entrada
116 if(key == 'A' || key == 'B' || key == 'C' || key == 'D'){
117     if(key == 'A'){    //Play
118         repFlag = 1;
119     }else if(key == 'B'){    //Pause
120         repFlag = 0;
121     }else if(key == 'C'){    //Next
122         repFlag = 2;
123     }else if(key == 'D'){    //Past
124         repFlag = 3;
125     }
126 }

204 if(repFlag != -1){
205     if(Firebase.RTDB.setInt(&fbdo, "music1/flag", repFlag)){
206         Serial.println("PASSED");
207         Serial.println("PATH: " + fbdo.dataPath());
208     }else{
209         Serial.println("FAILED");
210     }
211     repFlag = -1;
212     inputFlag = 0;
213 }
```

RASPBERRY

```
#Se escribe PAUSA en la Base de Datos
db.reference("/music1/flag").set(0)

#Funcion del hilo. Monitoreo de Firebase
def monitoreo(player):
    while True:
        global num
        # Monitoreo del numero de cancion
        num = db.reference("/music1/num").get()
        if num != -1:
            if 1 <= num <= 100:
                player.actualizarNum()
        # Monitoreo del estado del reproductor
        flag = db.reference("/music1/flag").get()
        if flag == 2:
            player.siguiente_mp3()
            player.reproducir_mp3()
        elif flag == 3:
            player.anterior_mp3()
            player.reproducir_mp3()
        elif flag == 1:
            player.reproducir_mp3()
        elif flag == 0:
            player.detener_mp3()
        time.sleep(0.5)
```

CONTROL DEL MBOT CON EL TECLADO MATRICIAL

ESP32

mBot

```
146 if(key == '*'){
147     key = 'F';
148     digitalWrite(21, HIGH);
149
150     while(key != '*'){
151         key = keypad.getKey();
152
153         if((key == '2') || (key == '8') || (key == '4') || (key == '6') || (key == '1') || (key == '3') || (key == '0')){
154             mbotControl = key - '0';
155             if(mbotControl == 0 ){
156                 mbotFlag = 7;
157             }else if(mbotControl == 1 ){
158                 mbotFlag = 5;
159             }else if(mbotControl == 2 ){
160                 mbotFlag = 1;
161             }else if(mbotControl == 3 ){
162                 mbotFlag = 6;
163             }else if(mbotControl == 4 ){
164                 mbotFlag = 3;
165             }else if(mbotControl == 6 ){
166                 mbotFlag = 4;
167             }else if(mbotControl == 8 ){
168                 mbotFlag = 2;
169             }
170             mbotFlag = -1;
171         }
172     }
173     digitalWrite(21, LOW);    //Apagado de LED rojo para indicar salida del modo de control manual
174 }
```

```
83 void vUSARTTask(void *pvParameters){
84     char valueReceived;
85     BaseType_t qStatus;
86     const TickType_t xTicksToWait = pdMS_TO_TICKS(100);
87     while(1){
88         qStatus = xQueueReceiveFromISR(UARTqueue, &valueReceived, pdTRUE);
89         if(qStatus == pdPASS){
90             if(valueReceived == 48){ //0
91                 stopM();
92             }else if(valueReceived == 49){ //1
93                 runF();
94                 _delay_ms(1000);
95                 stopM();
96             }else if(valueReceived == 50){ //2
97                 runB();
98                 _delay_ms(1000);
99                 stopM();
100            }else if(valueReceived == 51){ //3
101                runL();
102                _delay_ms(1000);
103                stopM();
104            }else if(valueReceived == 52){ //4
105                runR();
106                _delay_ms(1000);
107                stopM();
108            }else if(valueReceived == 53){ //5
109                rotateL();
110                _delay_ms(770);
111                stopM();
112            }else if(valueReceived == 54){ //6
113                rotateR();
114                _delay_ms(770);
115                stopM();
116            }else if(valueReceived == 55){ //7
117                danceFloor();
118                _delay_ms(4000);
119                stopM();
120            }
121        }
122        vTaskDelay(pdMS_TO_TICKS(100));
123    }
124 }
```

CONTROL DEL MBOT CON LA RASPBERRY

RASPBERRY

```
#Funcion del Boton Dance, manda un 7.  
def dance_Floor(self):  
    db.reference("/mbot1/flag").set(7)  
  
#Funcion del Boton Adelante, manda un 1.  
def mbot_forward(self):  
    db.reference("/mbot1/flag").set(1)  
  
#Funcion del Boton Atras, manda un 2.  
def mbot_backward(self):  
    db.reference("/mbot1/flag").set(2)  
  
#Funcion del Boton Izquierda, manda un 3.  
def mbot_left(self):  
    db.reference("/mbot1/flag").set(3)  
  
#Funcion del Boton Derecha, manda un 4.  
def mbot_right(self):  
    db.reference("/mbot1/flag").set(4)  
  
#Funcion del Boton Girar Izquierda, manda un 5.  
def girar_l(self):  
    db.reference("/mbot1/flag").set(5)  
  
#Funcion del Boton Girar Derecha, manda un 6.  
def girar_r(self):  
    db.reference("/mbot1/flag").set(6)
```

ESP32

```
215 if(Firebase.RTDB.getInt(&fbdo, "mbot1/flag")) {  
216     mbotFlag = fbdo.intData();  
217     if(mbotFlag != -1) {  
218  
219         Serial2.print(mbotFlag);  
220         if(Firebase.RTDB.setInt(&fbdo, "mbot1/flag", -1)) {  
221             Serial.println("PASSED");  
222             Serial.println("PATH: " + fbdo.dataPath());  
223         } else{  
224             Serial.println("FAILED");  
225         }  
226     }  
227     mbotFlag = -1;  
228 }
```

CONTROL DEL REPRODUCTOR CON SENsoRES

mBot

```
74 //Declaración de los pines de los sensores como entrada
75 DDRF &= ~(1 << DDF6); // Barrier Sensor Left (Pin 60)
76 DDRF &= ~(1 << DDF7); // Barrier Sensor Middle (Pin 61)
77 DDRK &= ~(1 << DDK0); // Barrier Sensor Right (Pin 62)
78 DDRK &= ~(1 << DDK3); // Collision Sensor Right (Pin 65)
79 DDRK &= ~(1 << DDK4); // Collision Sensor Left (Pin 66)
80 }
```

```
127 void vMBotSensorsTask(void *pvParameters){
128     TickType_t xLastWakeTIme;
129     while(1){
130         if (!(PINF & (1 << PINF6))){
131             sprintf(mybuffer, "\n3");
132             USART_Transmit_String((unsigned char *)mybuffer);
133         }
134         if (!(PINF & (1 << PINF7))){
135             sprintf(mybuffer, "\n18");
136             USART_Transmit_String((unsigned char *)mybuffer);
137         }
138         if !(PINK & (1 << PINK0))){
139             sprintf(mybuffer, "\n2");
140             USART_Transmit_String((unsigned char *)mybuffer);
141         }
142         if !(PINK & (1 << PINK3)){
143             sprintf(mybuffer, "\n0");
144             USART_Transmit_String((unsigned char *)mybuffer);
145         }
146         if !(PINK & (1 << PINK4)){
147             sprintf(mybuffer, "\n1");
148             USART_Transmit_String((unsigned char *)mybuffer);
149         }
150         xTaskDelayUntil(&xLastWakeTIme, (500/portTICK_PERIOD_MS)); //Retardo para tarea periodica
151     }
152 }
```

ESP32

```
244 if((Serial2.available() > 0)){ //Comprueba si hay datos nuevos disponibles en Serial2
245     Serial.println(69);
246     if(inputFlag == 0){
247         //Serial.print("Sensor: ");
248         uartInput = Serial2.readString(); //Lectura del dato
249         uartInput.trim(); //Filtra el dato, removiendo cualquier \r \n al final de la lectura
250         Serial.println(uartInput);
251
252         intValue = uartInput.toInt(); //Conversión del dato recibido a una variable entera
253         Serial.print("Converted to int: ");
254         Serial.println(intValue);
255
256         if(intValue <=3 ){
257             repFlag = intValue;
258         }else{
259             musicNum1 = random(1,100);
260         }
261         inputFlag = 1;
262     }
263 }
```

CONTROL DEL REPRODUCTOR CON LA INTERFAZ

RASPBERRY

```
#Libreria para el sonido
import pygame

#Libreria para la interfaz
from PyQt5 import QtWidgets, QtCore, QtGui

#Libreria para leer la carpeta
import os

#Libreria para leer metadatos de .mp3
from mutagen.mp3 import MP3
from mutagen.id3 import ID3

#Libreria para acceder a Firebase
import firebase_admin
from firebase_admin import db, credentials

#Libreria para delays
import time

#Libreria para ejecucion simultanea
import threading

#Libreria de la OLED
from luma.core.interface.serial import i2c
from luma.oled.device import ssd1306
from PIL import Image, ImageDraw, ImageFont

#Libreria para control del sistema
import sys
```

```
#Funcion que inicializa las partes de la interfaz.
def initUI(self):
    self.setWindowTitle('Reproductor MP3')
    #Se crea el layout principal y se divide
    main_layout = QtWidgets.QHBoxLayout()
    #Se crea el layout de la parte Izquierda.
    music_layout = QtWidgets.QVBoxLayout()

    self.boton_inicio = QtWidgets.QPushButton('Inicio', self)
    self.boton_inicio.clicked.connect(self.iniciar_monitoreo)
    music_layout.addWidget(self.boton_inicio)

    self.label = QtWidgets.QLabel('Detenido', self)
    self.label.setFont(QtGui.QFont('Helvetica', 12))
    self.label.setAlignment(QtCore.Qt.AlignCenter)
    music_layout.addWidget(self.label)

    self.label2 = QtWidgets.QLabel('', self)
    self.label2.setFont(QtGui.QFont('Helvetica', 12))
    self.label2.setAlignment(QtCore.Qt.AlignCenter)
    music_layout.addWidget(self.label2)
    self.actualizar_OLED('Detenido', '')

    self.input_num = QtWidgets.QLineEdit()
    self.input_num.returnPressed.connect(self.procesar_input)
    music_layout.addWidget(QtWidgets.QLabel("Número de Canción: "))
    music_layout.addWidget(self.input_num)

    #Se crea el layout de la parte Derecha.
    control_layout = QtWidgets.QHBoxLayout()

    self.boton_anterior = QtWidgets.QPushButton(self)
    self.update_prev_button()
    self.boton_anterior.clicked.connect(self.anterior_mp3)
    control_layout.addWidget(self.boton_anterior)

    self.boton_reproducir = QtWidgets.QPushButton(self)
    self.update_play_button()
    self.boton_reproducir.clicked.connect(self.toggle_b)
    control_layout.addWidget(self.boton_reproducir)

    self.boton_siguiente = QtWidgets.QPushButton(self)
    self.update_next_button()
    self.boton_siguiente.clicked.connect(self.siguiente_mp3)
    control_layout.addWidget(self.boton_siguiente)
```



**MUCHAS
GRACIAS**