Sketchs de ejemplo para el ponchitoCIII

Gonzalo F. Perez Paina (CIII-UTN-FRC)

18 de mayo de 2018

1. Introducción

El *ponchitoCIII* es una placa de expansión para Arduino o Intel Galileo que cuenta con: LEDs conectados a las salidas digitales, pulsadores conectado a entradas digitales, y un potenciometro conectado a la entrada del conversor analógico a digital.

Aquí se muestran algunos `skecth` de Arduino y su explicación para ser utilizado con el ponchitoCIII.

Este documento, junto a otro con detalles sobre el ponchitoCIII, y algunos sketchs Arduino de ejemplos para evaluar la placa se encuentran disponibles en https://github.com/ciiiutnfrc/ponchitoCIII.

2. Sketchs ejempos del IDE Arduino

Encendido y apagado de un LED.

En `File->Examples->01.Basics->Blink` se muestra un sketch de ejemplo que permite encender y apagar de forma continuada el LED incluido (build-in) en la placa Arduino, el cual está conectado al pin de entrada/salida digital nro. 13. Modificando este pin se puede hacer que encieda y apague un LED del ponchitoCIII. Para esto es necesario cambiar el valor de la constante simbólica `LED_BUILDIN` por el valor del nuevo pin (5, 6, o 9).

Cambio de intensidad de un LED.

En `File->Examples->01.Basics->Fade` se muestra un sketch que permite modificar el nivel de intensidad de la luz de un LED conectado al pin 9, que en el caso del ponchitoCIII se corresponde al de color rojo. El pin 9 entre otros puede actuar como salida de PWM (Pulse Width Modulation) y variar el ancho del pulso mediante la función `analogWrite()` lo cual modifica el nivel de intensidad el LED. Dichos pines están indicados en la placa con el símbolo "$\sim$" al lado del número de pin (tal como ~ 9). Modificando el valor de la variable `led` se puede cambiar la intensidad de otro LED del ponchitoCIII.

Lectura de valor analógico y envío por puerto serie.

En `File->Examples->01.Basics->AnalogReadSerial` se muestra un sketch de ejemplo en el cual se lee el valor de tensión presente en la entrada analógica `A0`, y se muestra su valor en la terminal serial del IDE Arduino. La terminal serial se abre desde `Tool->Serial Monitor`. El valor leído toma valores entre 0 y $2^{10} - 1 = 1023$, para el valor de tensión de entrada entre 0V y 5V. El sketch `File->Examples->01.Basics->ReadAnalogVoltage` muestra el valor directamente en voltios entre 0.00 y 5.00.

Listado 1: Enciende y apaga todos los LEDs.

```
/*
 * Enciende y apaga todos los LEDs
 * (basado en Examples/01.Basics/Blink)
 *
 * El código de este ejemplo es de dominio público.
 */

#define GPIO_LED_VERDE 5
#define GPIO_LED_AMARILLO 6
#define GPIO_LED_ROJO 9

#define LED_ENCENDIDO HIGH
#define LED_APAGADO LOW

// La función 'setup' se ejecuta una única vez al presionar reset o
    encender la placa.
void setup() {
  // inicialización de los pines GPIO como salida para los LEDs.
  pinMode(GPIO_LED_VERDE, OUTPUT);
  pinMode(GPIO_LED_AMARILLO, OUTPUT);
  pinMode(GPIO_LED_ROJO, OUTPUT);
}

// La función 'loop' corre indefinidamente una y otra vez.
void loop() {
  // Enciende todos los LEDs.
  digitalWrite(GPIO_LED_VERDE, LED_ENCENDIDO);
  digitalWrite(GPIO_LED_AMARILLO, LED_ENCENDIDO);
  digitalWrite(GPIO_LED_ROJO, LED_ENCENDIDO);

  // Espera 1 seg.
  delay(1000);

  // Enciende todos los LEDs.
  digitalWrite(GPIO_LED_VERDE, LED_APAGADO);
  digitalWrite(GPIO_LED_AMARILLO, LED_APAGADO);
  digitalWrite(GPIO_LED_ROJO, LED_APAGADO);

  // Espera 1 seg.
  delay(1000);
}
```

3. Sketch del ponchitoCIII

3.1. Salidas digitales – LEDs

3.2. Entradas digitales – pulsadores

3.3. Entrada analógica – Conversor analógico a digital

Referencias

[1] IDE Arduino. https://www.arduino.cc/en/Main/Software

LaTeX / Creado en abril de 2018

Listado 2: Conmuta el estado de los LEDs a través del puerto serie.

```
/*
 * Recibe una letra por puerto serie para cambiar el estado de los LEDS.
 * 'v': cambia el estado del LED verde.
 * 'a': cambia el estado del LED amarillo.
 * 'r': cambia el estado del LED rojo.
 *
 * Se puede probar con el "Monitor serie" del IDE Arduino.
 *
 * El código de este ejemplo es de dominio público.
 */

#define GPIO_LED_VERDE    5
#define GPIO_LED_AMARILLO 6
#define GPIO_LED_ROJO     9

 // La función 'setup' se ejecuta una única vez al presionar reset o
     encender la placa.
void setup() {
  // Inicializa el puerto serie (UART) a 9600 bps.
  Serial.begin(9600);

  // inicialización de los pines GPIO como salida para los LEDs.
  pinMode(GPIO_LED_VERDE, OUTPUT);
  pinMode(GPIO_LED_AMARILLO, OUTPUT);
  pinMode(GPIO_LED_ROJO, OUTPUT);
}

// La función 'loop' corre indefinidamente una y otra vez.
void loop() {
  size_t n;
  uint8_t letra[1];

  // Lee la letra (1 byte) por puerto serie.
  n = Serial.readBytes(letra, 1);
  if(n == 1)
  {
    switch(letra[0])
    {
      case 'v':
        toggle(GPIO_LED_VERDE);
        break;
      case 'a':
        toggle(GPIO_LED_AMARILLO);
        break;
      case 'r':
        toggle(GPIO_LED_ROJO);
        break;
    }
  }
}

// Función para cambiar el estado de un LED (GPIO).
void toggle(uint8_t gpio)
{
  if(digitalRead(gpio) == HIGH)
    digitalWrite(gpio, LOW);
  else
    digitalWrite(gpio, HIGH);
}
```

Listado 3: Muestra por el monitor serie el estado de los pulsadores en binario.

```
/*
 * Envía los estados de las entradas (pulsadores) por puerto serie (UART).
 * Se puede probar con el "Monitor serie" del IDE Arduino.
 *
 * El código de este ejemplo es de dominio público.
 */

#define GPIO_BOTON_IZQ 8
#define GPIO_BOTON_MED 7
#define GPIO_BOTON_DER 4

// La función 'setup' se ejecuta una única vez al presionar reset o
    encender la placa.
void setup() {
  // Inicializa el puerto serie (UART) a 9600 bps.
  Serial.begin(9600);

  // Configura los GPIO de los pulsadores como entradas.
  pinMode(GPIO_BOTON_IZQ, INPUT);
  pinMode(GPIO_BOTON_MED, INPUT);
  pinMode(GPIO_BOTON_DER, INPUT);
}

// La función 'loop' corre indefinidamente una y otra vez.
void loop() {
  // Lee el valor de los pines de entrada (GPIO).
  int boton_izq = digitalRead(GPIO_BOTON_IZQ);
  int boton_med = digitalRead(GPIO_BOTON_MED);
  int boton_der = digitalRead(GPIO_BOTON_DER);

  // Envía por puerto serie el estado del pulsador.
  Serial.write(boton_izq + '0');
  Serial.write(boton_med + '0');
  Serial.write(boton_der + '0');
  Serial.write('\n');

  delay(100);
}
```

Listado 4: Muestra por el monitor serie el estado de los pulsadores en texto.

```
/*
 * Envía los estados de los pulsadores por puerto serie (UART).
 * Se puede probar con el "Monitor serie" del IDE Arduino.
 *
 * El código de este ejemplo es de dominio público.
 */

#define GPIO_BOTON_IZQ 8
#define GPIO_BOTON_MED 7
#define GPIO_BOTON_DER 4

#define ESTADO_PRESIONADO 0

// La función 'setup' se ejecuta una única vez al presionar reset o
    encender la placa.
void setup() {
  // Inicializa el puerto serie (UART) a 9600 bps.
  Serial.begin(9600);

  // Configura los GPIO de los pulsadores como entradas.
  pinMode(GPIO_BOTON_IZQ, INPUT);
  pinMode(GPIO_BOTON_MED, INPUT);
  pinMode(GPIO_BOTON_DER, INPUT);
}

// La función 'loop' corre indefinidamente una y otra vez.
void loop() {
  // Lee el valor de los pines de entrada (GPIO).
  int boton_izq = digitalRead(GPIO_BOTON_IZQ);
  int boton_med = digitalRead(GPIO_BOTON_MED);
  int boton_der = digitalRead(GPIO_BOTON_DER);

  // Envía por puerto serie el estado del pulsador.
  Serial.print("Estado izq.: ");
  Serial.println(boton_izq == ESTADO_PRESIONADO ? "presionado" : "libre");
  Serial.print("Estado med.: ");
  Serial.println(boton_med == ESTADO_PRESIONADO ? "presionado" : "libre");
  Serial.print("Estado der.: ");
  Serial.println(boton_der == ESTADO_PRESIONADO ? "presionado" : "libre");

  Serial.println("");
  delay(1000);
}
```

Listado 5: ADCFull.

```
/*
 * Envia por puerto serie el valor del ADC en diferentes formatos por
     petición.
 * 'c': envía el valor entero como cadena ("0000" a "1023").
 * 'e': envía el valor como nro. entero.
 * 'f': envía el valor del voltaje como nro. de punto flotante (float).
 *
 * Se puede probar con el "Monitor serie" del IDE Arduino.
 *
 * El código de este ejemplo es de dominio público.
 */

#define CANAL_ADC A0

// Union de entero y unsigned char.
typedef union
{
  uint8_t uc[4];
  int i;
} int_uc_t;

// Union de float y unsigned char.
typedef union
{
  uint8_t uc[4];
  float f;
} float_uc_t;

// Variables globales.
size_t n;
uint8_t solicitud;

char cadena[5];   // Valor del ADC como cadena.
int_uc_t i_uc;    // Valor del ADC en entero.
float_uc_t f_uc;  // Valor del ADC en punto flotante (float).

// La función 'setup' se ejecuta una única vez al presionar reset o
    encender la placa.
void setup() {
  // Inicializa el puerto serie (UART) a 9600 bps.
  Serial.begin(9600);
}

// La función 'loop' corre indefinidamente una y otra vez.
void loop() {
  // Lectura de la entrada analógica.
  int valor_adc = analogRead(CANAL_ADC);

  // Lee la letra (1 byte) por puerto serie.
  n = Serial.readBytes(&solicitud, 1);

  if(n == 1)
  {
    switch(solicitud)
    {
      // Envia el valor del ADC como cadena.
      case 'c':
        entero_a_cadena(valor_adc, 4, cadena);
        Serial.write(cadena, 5);
        break;
```

Listado 6: ADCFull (cont.).

```
59        case 'e':
60          i_uc.i = (int)valor_adc;
61          Serial.write(i_uc.uc, 4);
62          break;
63        case 'f':
64          f_uc.f = valor_adc * (5.0 / 1023.0);
65          Serial.write(f_uc.uc, 4);
66          break;
67      }
68    }
69
70    delay(100);
71  }
72
73  void entero_a_cadena(unsigned int entero, unsigned char cant_digitos, char
        * cadena)
74  {
75    unsigned int n, res;
76
77    for(n = 0; n < cant_digitos; n++)
78    {
79      res = (entero / potencia_entero(10, cant_digitos-n-1));
80      cadena[n] = res + '0';
81      entero -= res * potencia_entero(10, cant_digitos-n-1);
82    }
83    cadena[cant_digitos] = '\0';
84  }
85
86  int potencia_entero(int x, int n)
87  {
88    if(n == 0)
89      return 1;
90
91    int r = 1;
92    while(n--)
93      r *= x;
94
95    return r;
96  }
```