URJC Escuela Técnica Superior de Ingeniería de Telecomunicación

GSyC

Sistemas Empotrados y de Tiempo Real

Desarrollo con Arduino MultiTasking

Grado en Ingeniería de Robótica Software

Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación

Roberto Calvo Palomino
roberto.calvo@urjc.es

MultiTasking

MultiTasking

Sketch

- Los sketch en Arduino son simples y fáciles de utilizar al principio cuando los ejemplos son muy básicos
  - Enceder/apagar un led.
  - Leer un pulsador y encender un led
- Pero la complejidad aumenta cuando el número de tareas a realizar es alta y cada una necesita una periodicidad diferente.
  - Led intermitente cada 500 ms
  - Leer entrada digital pulsador cada 170 ms
  - Leer sensor temperatura cada segundo
  - ….

Scketch

- Posible implementación con millis()

```
void loop() {

  /* Updates frequently */
  unsigned long currentTime = millis();

  /* This is the event */
  if (currentTime - previousTime >= eventInterval1) {
    /* Event code */
    Serial.println("Task1");
   /* Update the timing for the next time around */
    previousTime1 = currentTime;
  }
  if (currentTime - previousTime >= eventInterval2) {
    /* Event code */
    Serial.println("Task2");
   /* Update the timing for the next time around */
    previousTime2 = currentTime;
  }
  if (currentTime - previousTime >= eventInterval3) {
    /* Event code */
    Serial.println("Task3");
   /* Update the timing for the next time around */
    previousTime3 = currentTime;
  }
}
```

Scketch

- Posible implement ción con millis()

```
void l

  /* Up                ntly */
  unsigi            entTime

  /* This                  */
  if (curr                  ventInterval
    /* Even
    Serial.p
   /* Update                ext time around
    previousTi
  }
  if (currentTi                 >= eventInterval
    /* Event co
    Serial.pri
   /* Update                   round */
    previou
  }
  if (cu              previousTi
    /*                */
    Se                tln("Task3");
   /*                the timing for the next time around */
                    usTime3 = currentTime;
  }
}
```

ArduinoThread

- Arduino incorpora una librería que nos permite ejecutar “threads” en Arduino.
- Compatible con todas las arquitecturas Arduino
- No ejecuta threads en paralelo
- Planificar y manejar fácilmente tareas periódicas
- Definir tiempos fijos o variables entre ejecuciones
- Organizar el código limpiamente:
  - Todas las lecturas de sensores en un thread.
  - Mantiene el bucle principal limpio.
- Esconde la complejidad de cada thread.

ArduinoThread

- Documentación:
  - https://www.arduino.cc/reference/en/libraries/arduinothread/
  - https://github.com/ivanseidel/ArduinoThread

ArduinoThread

- Añadir las librerías necesarias.

```
#include <Thread.h>
#include <StaticThreadController.h>
#include <ThreadController.h>
```

- Creamos un thread asignándole una periodicidad y callback

```
  Thread myThread = Thread();
  myThread.enabled = true;
  myThread.setInterval(300);
  myThread.onRun(callback_thread1);
```

- El callback puede implementar cualquier código

```
void callback_thread1() {
  Serial.println("Hello! " + String(millis()));
}
```

ArduinoThread

```
#include <Thread.h>

void callback_thread1() {
  Serial.println("Hello! " + String(millis()));
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(100);

  myThread.enabled = true;
  myThread.setInterval(300);
  myThread.onRun(callback_thread1);

}

void loop() {

  // Check si podemos ejecutar el thread
  if(myThread.shouldRun()){
    myThread.run();
  }
}
```

ArduinoThread: Controller

- ¿Y qué ocurre si tenemos 20 threads distintos?
- Podemos usar el controlador de ArduinoThread
- El controlador permite manejar, activar y ejecutar numerosos threads al mismo tiempo.

ArduinoThread: Controller

```
#include <Thread.h>
#include <StaticThreadController.h>
#include <ThreadController.h>

ThreadController controller = ThreadController();
Thread myThread = Thread();

void callback_thread1() {
  Serial.println("Hello! " + String(millis()));
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(100);

  myThread.enabled = true;
  myThread.setInterval(300);
  myThread.onRun(callback_thread1);

  controller.add(&myThread);
}

void loop() {

  controller.run();

}
```

Añadimos las librerías

Instanciamos el controlador

Añadimos el thread al controlador

El bucle principal solo llama al controlador en cada iteración

ArduinoThread

¿Cómo generamos un thread que permita parpadear un LED sin que utilice más recursos de los necesarios utilizando ArduinoThread?

ArduinoThread

```
class LedThread: public Thread {

public:
  int pin;
  bool state;

  LedThread(int _pin): Thread() {
    pin = _pin;
    state = true;

    pinMode(pin, OUTPUT);
  }

  bool shouldRun(unsigned long time){
    return Thread::shouldRun(time);
  }

  void run(){

    Thread::run();

    digitalWrite(pin, state ? HIGH : LOW);
    state = !state;

  }
};
```

Heredamos de Thread

Constructor recibe el pin del LED

shouldRun es llamado siempre antes de run(). Solo ejecutará run() si esta función devuelve TRUE

Run(), ejecuta el código deseado de este Thread. Siempre hay que llamar al padre Thread::run()

ArduinoThread

```
#define PIN_LED 8

ThreadController controller = ThreadController();

void setup() {

  Serial.begin(9600);
  delay(100);

  LedThread* ledThread = new LedThread(LED_BUILTIN);
  ledThread->setInterval(1000);
  controller.add(ledThread);

  LedThread* ledThread2 = new LedThread(PIN_LED);
  ledThread2->setInterval(2000);
  controller.add(ledThread2);

}
```

Primer objeto LedThread con intervalo de 1 segundo y LED_BUILTIN

Segundo objeto LedThread con intervalo de 2 segundos y PIN_LED = 8

ArduinoThread

- Primer aproximación a multi-tarea en Arduino mono-core
- Nos permite programar y organizar la funcionalidad de nuestro sketch en diferentes hilos/threads de ejecución.
- Si un thread se queda bloqueado, todo el programa se queda bloqueado.
- Nos permite programar en Arduino sin necesidad de delays.

URJC Escuela Técnica Superior de Ingeniería de Telecomunicación