

# Chibi OS

---

Moya Garduño Omar  
Cruz Reyes Antonio



# Requisitos de los SOTR

---

Los SOTR se caracterizan por presentar requisitos especiales en cinco áreas generales

- Determinismo
- Sensibilidad
- Control del usuario
- Fiabilidad
- Tolerancia a los fallos



# ¿Sistema Operativo en Tiempo Real?

---

- Este Sistema Operativo se hizo publico en septiembre de 2007
- Es un sistema informático que interacciona repetidamente con su entorno físico, responde a los estímulos que recibe del mismo dentro de un plazo de tiempo determinado.
- Esta diseñado para aplicaciones embebidas para microcontroladores de 8 16 y 32 bits



# ChibiOS/RT

---

ChibiOS / RT está diseñada para aplicaciones profundamente arraigadas en tiempo real, donde la eficiencia y la ejecución de código compacto son requisitos importantes. Este RTOS se caracteriza por su alta portabilidad, tamaño compacto (recordemos que ya tenemos consumido en ARDUINO unos 2 a 3k) y, sobre todo, por su arquitectura optimizada para la conmutación de contexto de procesos extremadamente eficiente.

Cuando usamos RTOS podemos crear procesos separados denominados hilos. Cada uno de los procesos se parece a la función loop independiente.

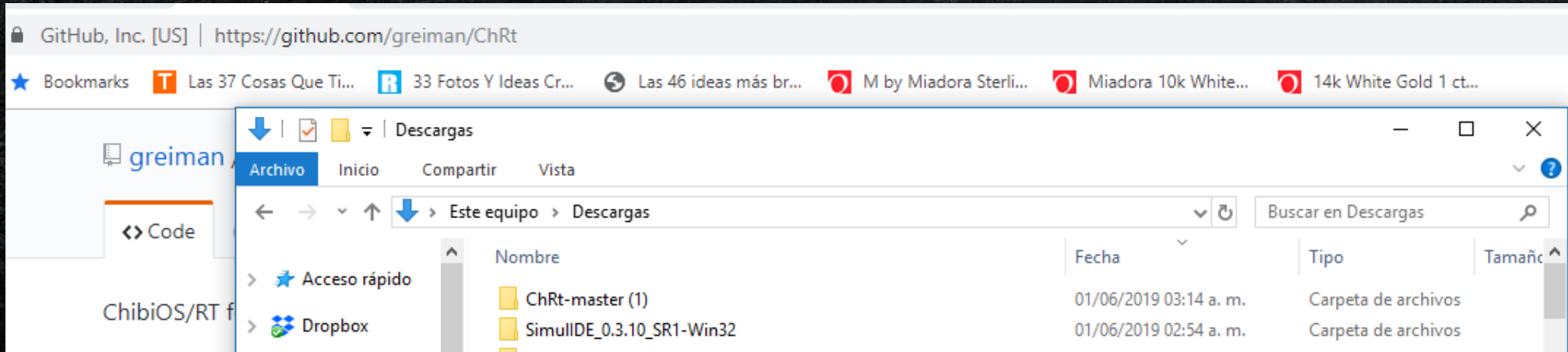
Se puede crear un programa con mucho bucles cada uno de ellos ejecutará su tarea independiente.



# Como instalarlo

El ChibiOS es una simple biblioteca que debe copiarse a la carpeta de biblioteca Arduinos. Y el programa es simple script Arduino creado en Arduino IDE, es decir una simple llamada `#include "ChibiOS_AVR.h"`.

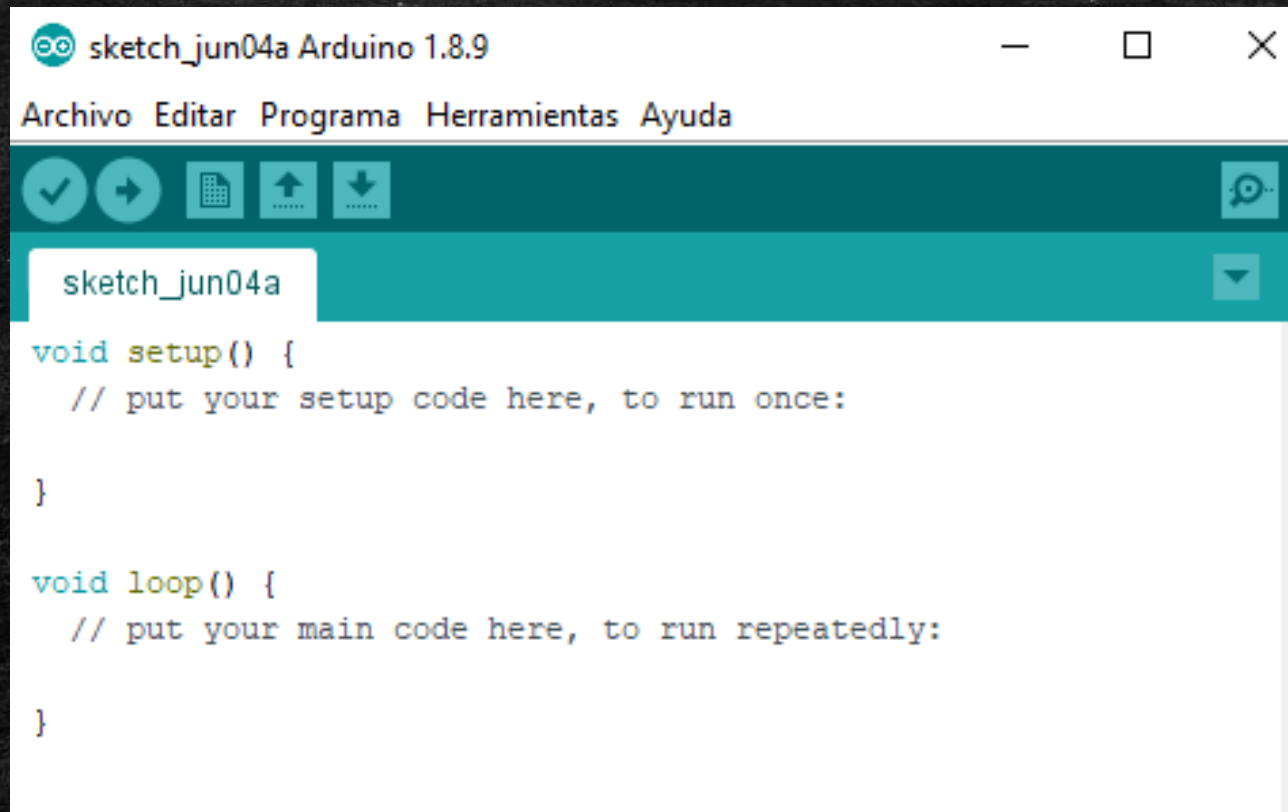
1. Nosotros ingresamos a la liga siguiente y descargamos la librería completa.  
<https://github.com/greiman/ChRt>





# Como instalarlo

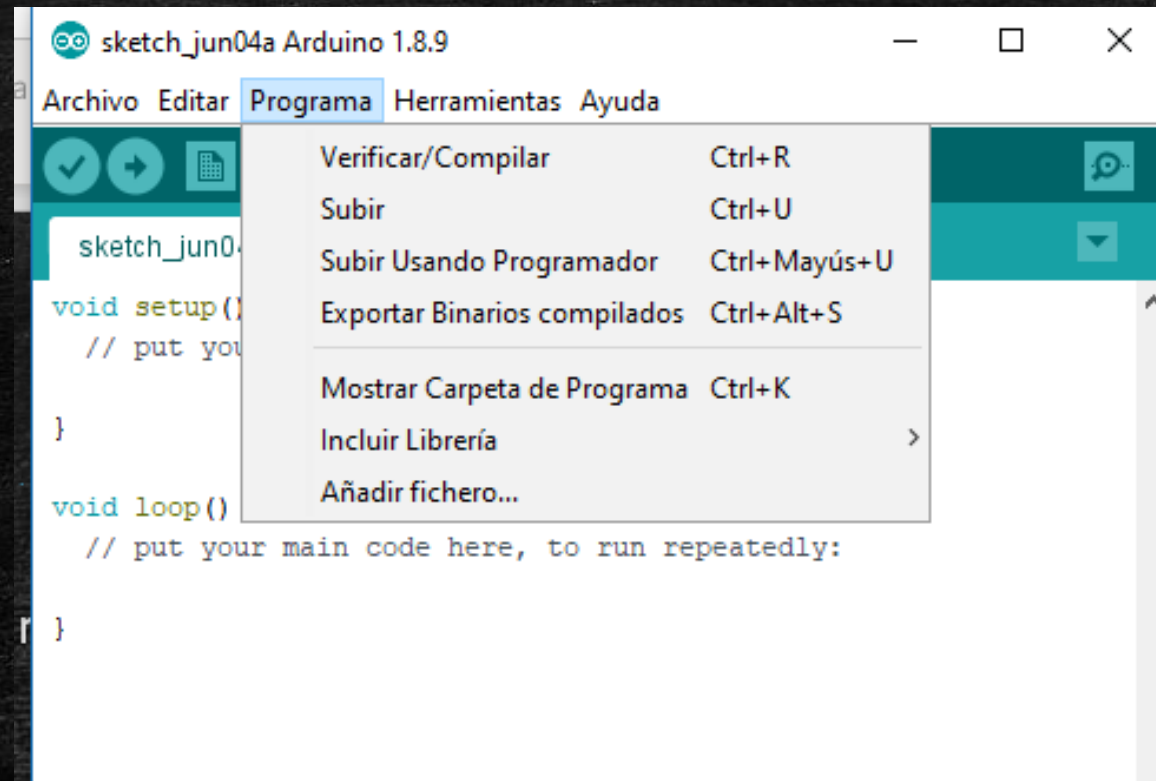
## 2. Ingresar a la aplicación de Arduino





# Como instalarlo

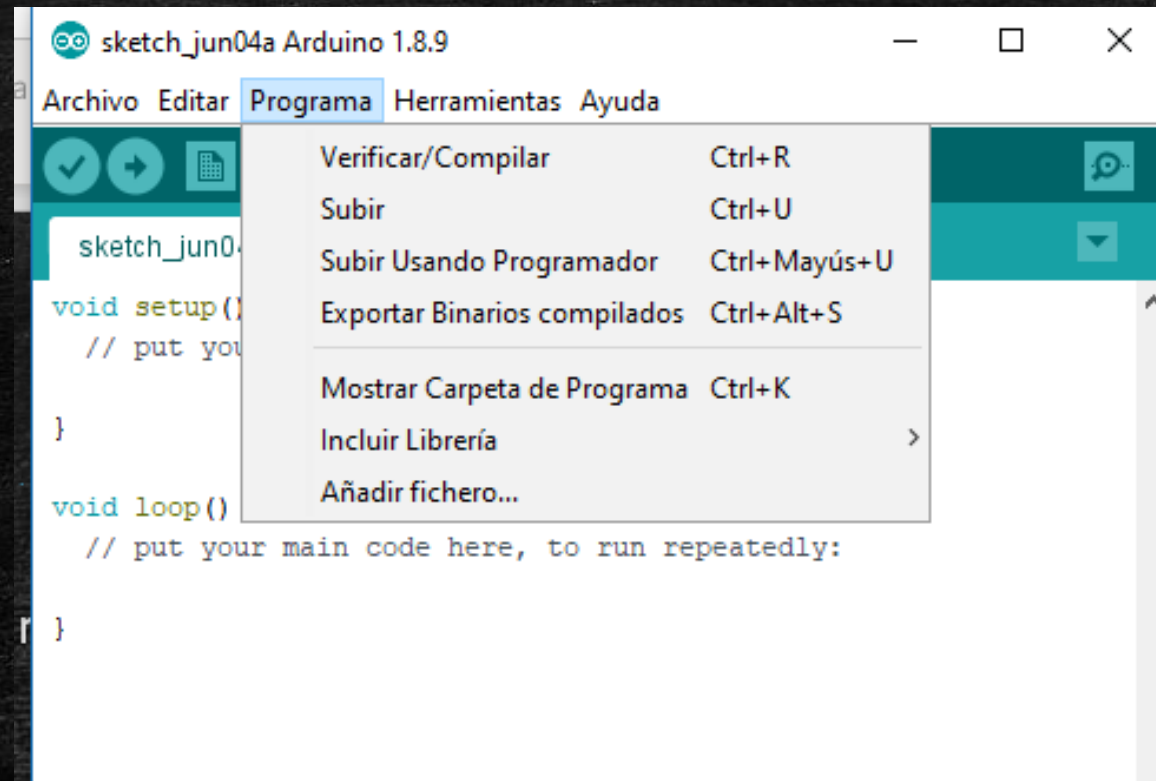
## 3. Ir al menú de programa





# Como instalarlo

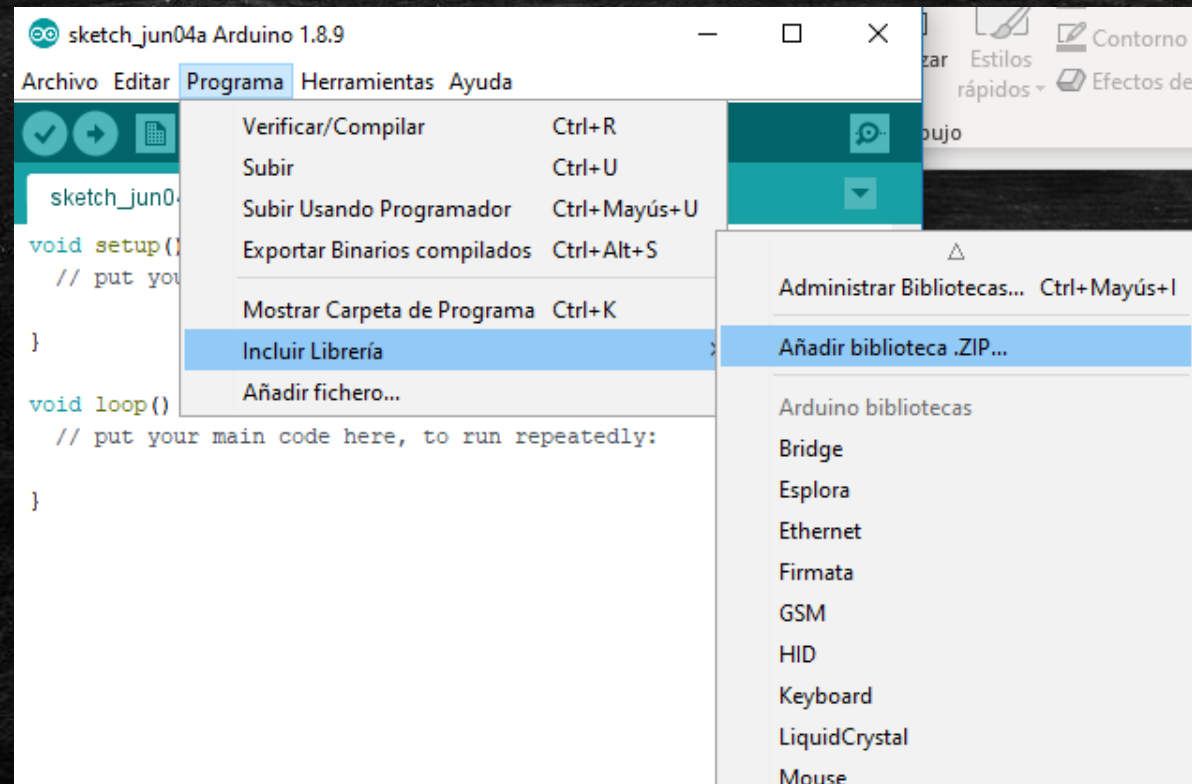
4. Ingresar a donde dice incluir librería.





# Como instalarlo

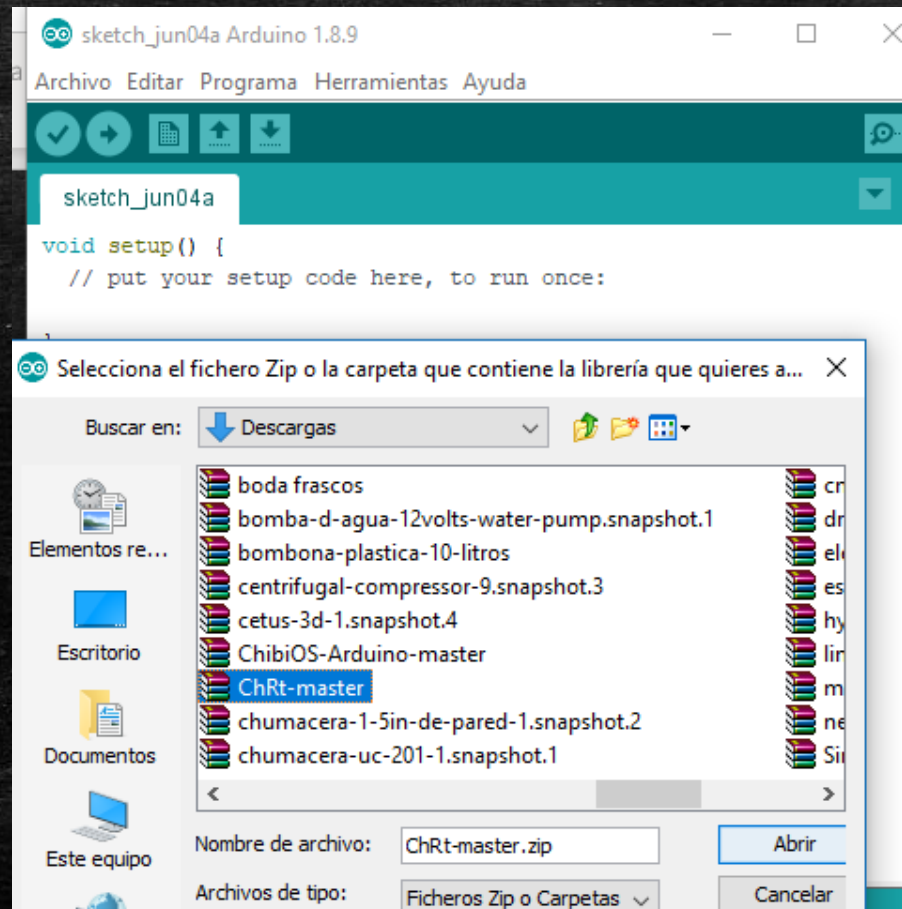
## 5. Seleccionar añadir biblioteca ZIP.





# Como instalarlo

6. Seleccionar la carpeta comprimida descargada.

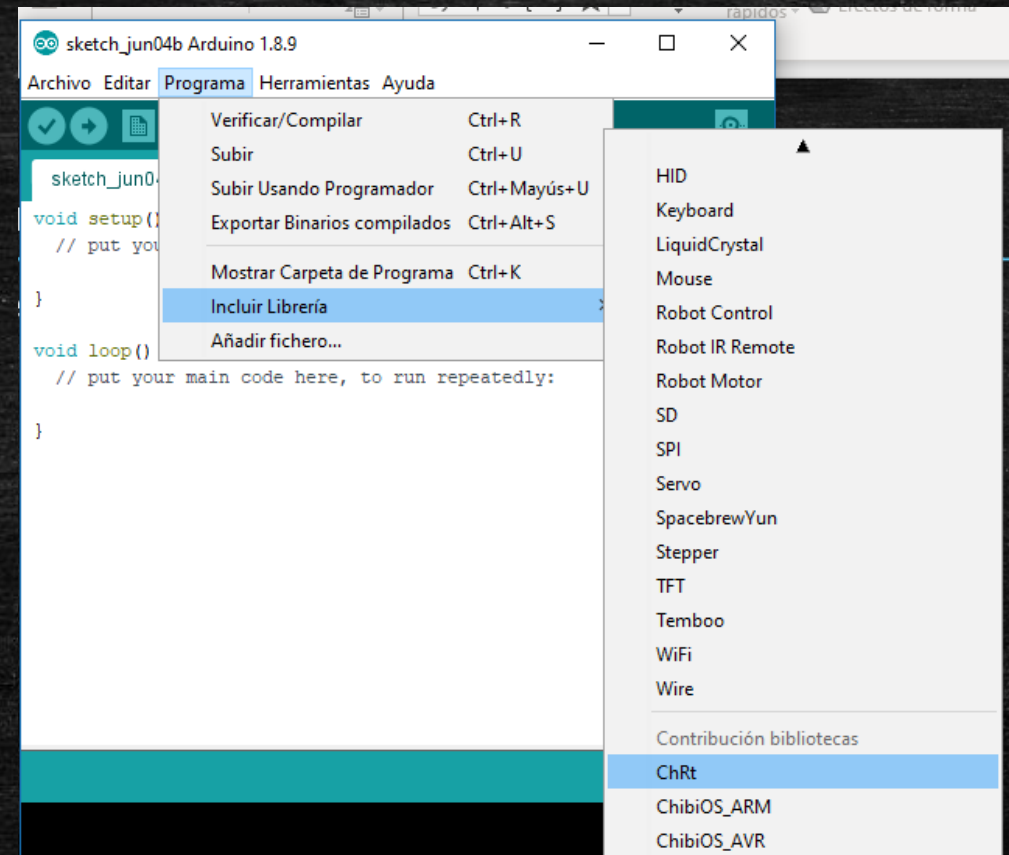




# Como instalarlo

7. Una vez terminada la instalación, cerrar y volver a abrir el programa.

Se va a comprobar que están 3 Librerías nuevas.





# Ejemplo Arduino 1

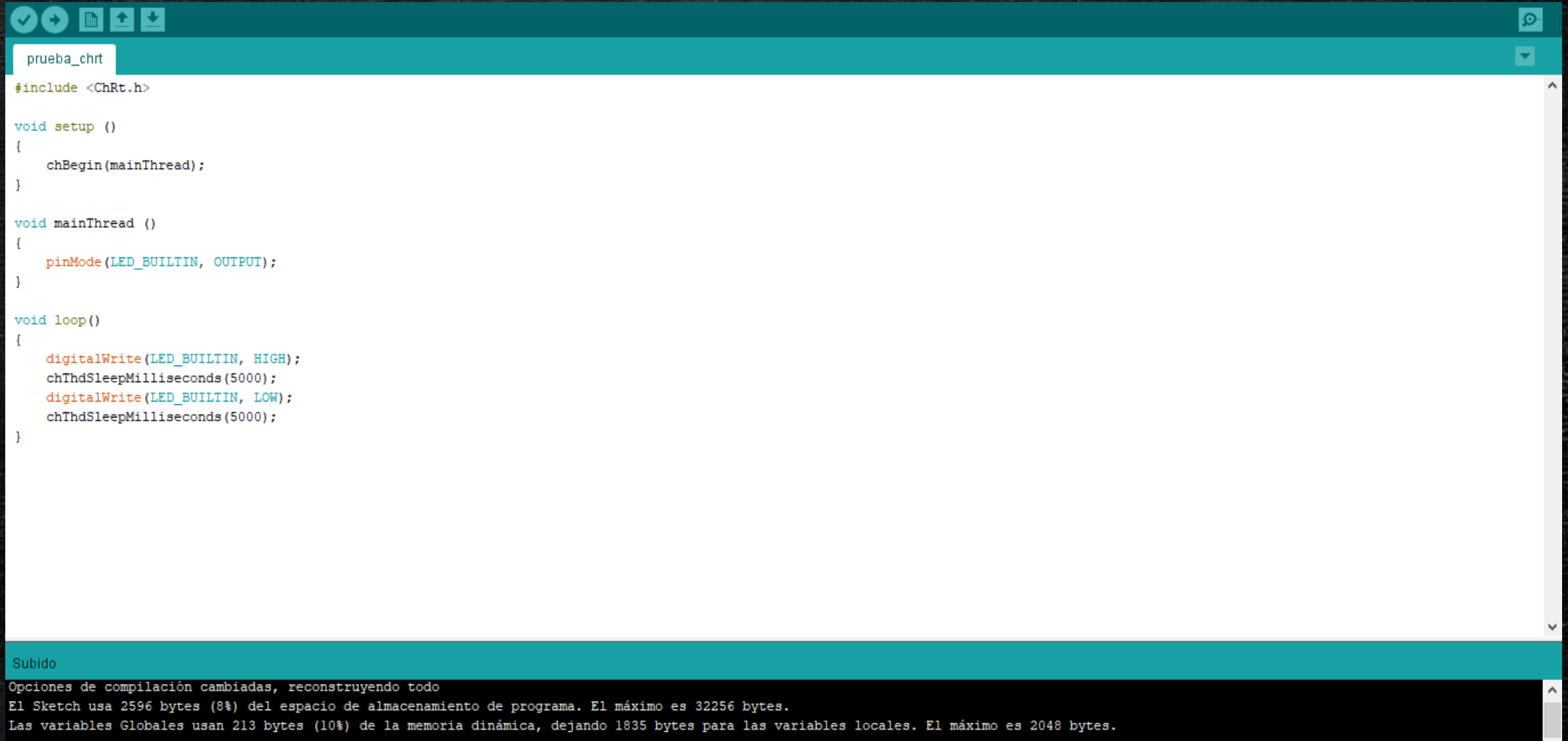
---

En el siguiente ejercicio vamos a poder visualizar un ejercicio muy sencillo en el cual el comando para que funcione la librería es: `#include <ChRt.h>` ya antes mencionado.

Y podemos visualizar en el ejercicio con Arduino como prende y apaga el foco led por la señal que se le mando de cada 5 segundos.



# Ejemplo Arduino 1



```
#include <ChRt.h>

void setup ()
{
    chBegin(mainThread);
}

void mainThread ()
{
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
    digitalWrite(LED_BUILTIN, HIGH);
    chThdSleepMilliseconds(5000);
    digitalWrite(LED_BUILTIN, LOW);
    chThdSleepMilliseconds(5000);
}
```

Subido

Opciones de compilación cambiadas, reconstruyendo todo

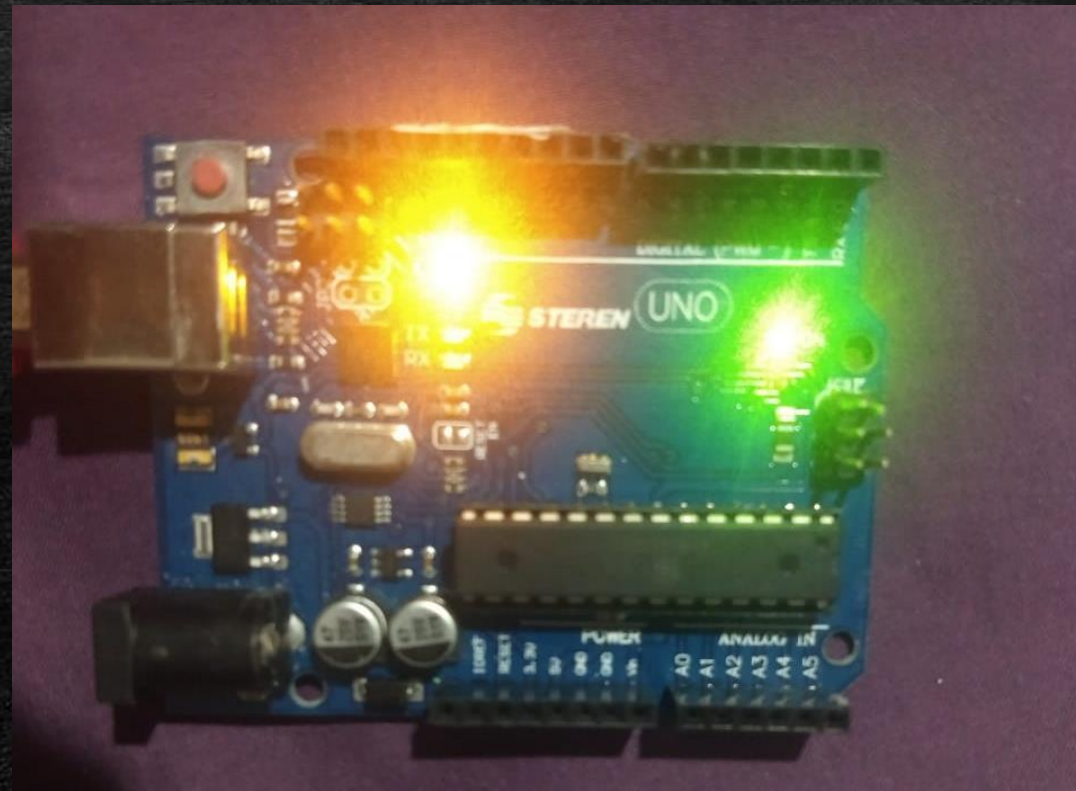
El Sketch usa 2596 bytes (8%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.

Las variables Globales usan 213 bytes (10%) de la memoria dinámica, dejando 1835 bytes para las variables locales. El máximo es 2048 bytes.



# Ejemplo Arduino 1

Arduino prendiendo y apagando cada 5 segundos





## Ejemplo Arduino 2

---

Hay diferentes ejemplos sobre programas con SORT y Arduino los cuales son necesario tener la tarjeta física para poder operarla, dado el punto que no todos operan en Arduino Uno y unos específicamente son de Due o en la tarjeta Tessy 3 no nos permitió probarlos pero los mencionamos por simple efecto de ejemplo.



```

// Example of synchronous messages
#include <ChibiOS_ARM.h>
// data structures and stack for thread 2
static THD_WORKING_AREA(waTh2, 1000);
thread_t *thread2Pointer;

// data structures and stack for thread 3
static THD_WORKING_AREA(waTh3, 1000);
thread_t *thread3Pointer;

// data structures and stack for thread 3
static THD_WORKING_AREA(waThHB, 64);

struct messageData{ //simple struct to hold message data
    int timeStamp;
    char* message;
};
static THD_FUNCTION(thdTLF1, arg) {
    messageData* myMessage; //create an empty pointer of type messageData
    static int lastTimeStamp = 0;

    while(1){
        chMsgWait(); //wait for a new message
        myMessage = (messageData*)chMsgGet(thread3Pointer); //set our message pointer to the
location of the message
        int receiveTime = chVTGetSystemTime(); //get the current tick

        Serial.println("Message Recieved:");
        Serial.print("Message: ");
        Serial.println(myMessage->message);

        chMsgRelease(thread3Pointer, (msg_t)&myMessage); //release the message with a pointer to
the original message as our ping back message
        lastTimeStamp = myMessage->timeStamp;
    }
}

```

```

static THD_FUNCTION(thdTLF2, arg) {
    messageData myMessage; //create a new message with type messageData

    while(1){
        myMessage.message = "Hello World"; //set the payload of the message (in this case, the name)
        myMessage.timeStamp = chVTGetSystemTime(); //and the timestamp in ticks just before sending
        chMsgSend(thread2Pointer, (msg_t)&myMessage); //send the message and wait for reply (synchronous)

        chThdSleep(100);
    }
}

//basic heartbeat thread to tell us whether weve crashed
static THD_FUNCTION(heartbeat, arg){
    int ledPin = 13;

    pinMode(ledPin, OUTPUT);
    while(1){
        digitalWrite(ledPin, HIGH);
        chThdSleep(10);
        digitalWrite(ledPin, LOW);
        chThdSleep(300);
        digitalWrite(ledPin, HIGH);
        chThdSleep(10);
        digitalWrite(ledPin, LOW);
        chThdSleep(1000);
    }
}

```



## Ejemplo Arduino 2

---

- El ejemplo anterior realiza la función de mandar un mensaje dentro del propio entorno de Chibi OS mientras que después de un cierto retardo y habiendo comprobado que una función recibió el mensaje se regresa otro mensaje como recibido.



# Ejemplo Arduino 3

```
1 // Example of using mutexes
2 #include <ChibiOS_ARM.h>
3
4 // data structures and stack for thread 2
5 static THD_WORKING_AREA(waTh2, 100);
6
7 // data structures and stack for thread 3
8 static THD_WORKING_AREA(waTh3, 100);
9
10 // data structures and stack for thread 3
11 static THD_WORKING_AREA(waThHB, 64);
12
13
14 //create a mutex that we will use to control access to the serial line
15 MUTEX_DECL(serialMtx);
16
17
18
19 static THD_FUNCTION(thdTLF1, arg) {
20
21     while(1){
22         chMtxLock(&serialMtx); //lock access to the serial line to only this thread
23
24         for(int i = 0; i < 10; i++){
25             Serial.println(i);
26             chThdSleep(100);
27         }
28
29         chMtxUnlock(&serialMtx); //release access to the serial line
30     }
```

```
37 static THD_FUNCTION (thdTLF2, arg) {
38
39     while(1){
40         chMtxLock(&serialMtx); //lock access to the serial line to only this thread
41
42         Serial.println("My Turn!");
43         chThdSleep(10);
44
45         chMtxUnlock(&serialMtx); //release access to the serial line
46     }
47
48 }
49
50
51
52 //basic heartbeat thread to tell us whether weve crashed
53 static THD_FUNCTION(heartbeat, arg){
54     int ledPin = 13;
55
56     pinMode(ledPin, OUTPUT);
57     while(1){
58         digitalWrite(ledPin, HIGH);
59         chThdSleep(10);
60         digitalWrite(ledPin, LOW);
61         chThdSleep(300);
62         digitalWrite(ledPin, HIGH);
63         chThdSleep(10);
64         digitalWrite(ledPin, LOW);
65         chThdSleep(1000);
66     }
67 }
```



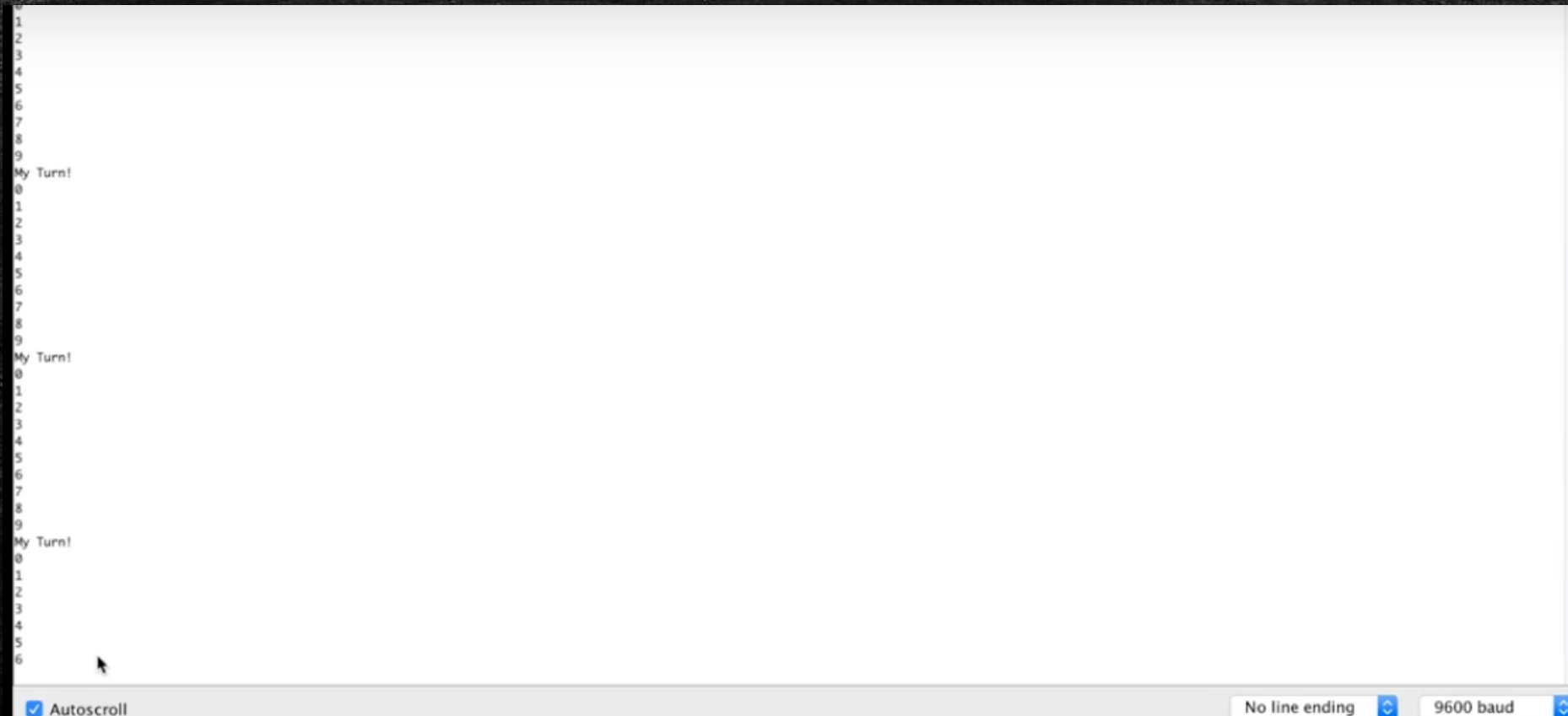
# Ejemplo Arduino 3

```
//-----  
void setup() {  
  Serial.begin(115200);  
  
  // initialize and start Chibios  
  chBegin(chSetup);  
  
  // should not return  
  while(1);  
}  
  
//-----  
//-----  
void chSetup() {  
  // schedule thread 2  
  chThdCreateStatic(waTh2, sizeof(waTh2), NORMALPRIO, thdTLF1, NULL);  
  
  // schedule thread 3  
  chThdCreateStatic(waTh3, sizeof(waTh3), NORMALPRIO + 1, thdTLF2, NULL);  
  
  // schedule heartbeat thread  
  chThdCreateStatic(waThHB, sizeof(waThHB), NORMALPRIO - 1, heartbeat, NULL);  
  
  while(1){  
    chThdSleep(1000);  
  }  
  
  // main thread is thread 1 at NORMALPRIO  
}  
//-----  
void loop() { /* not used */ }
```



# Ejemplo Arduino 3

En este ejemplo como resultado da las ordenes usando Mutex y prioridades, las cuales cada 10 segundos esta mostrando un mensaje el cual da el turno y sigue corriendo el programa, actualizándose en un Loop



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
My Turn!  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
My Turn!  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
My Turn!  
0  
1  
2  
3  
4  
5  
6
```

Autoscroll No line ending 9600 baud