REAL-TIME OPERATING SYSTEM ON ARDUINO

Nicolas Pantano

CONTENT

1. Real-Time Operating System
    a. Why do we need a RTOS ?
    b. Multitasking
        i. Cooperative
        ii. Preemptive
2. Demonstration application
    a. Arduino
    b. FreeRTOS
    c. Implementation
    d. Results
3. Conclusion

WHY DO WE NEED A RTOS?

We need a RTOS for two main reasons.

- Precise timing
- High degree of reliability

MULTITASKING

- Main feature of an operating system
- Goal ⟶ sharing access to the processor
- Multiple ways to do so
    - Cooperative multitasking
    - Preemptive multitasking

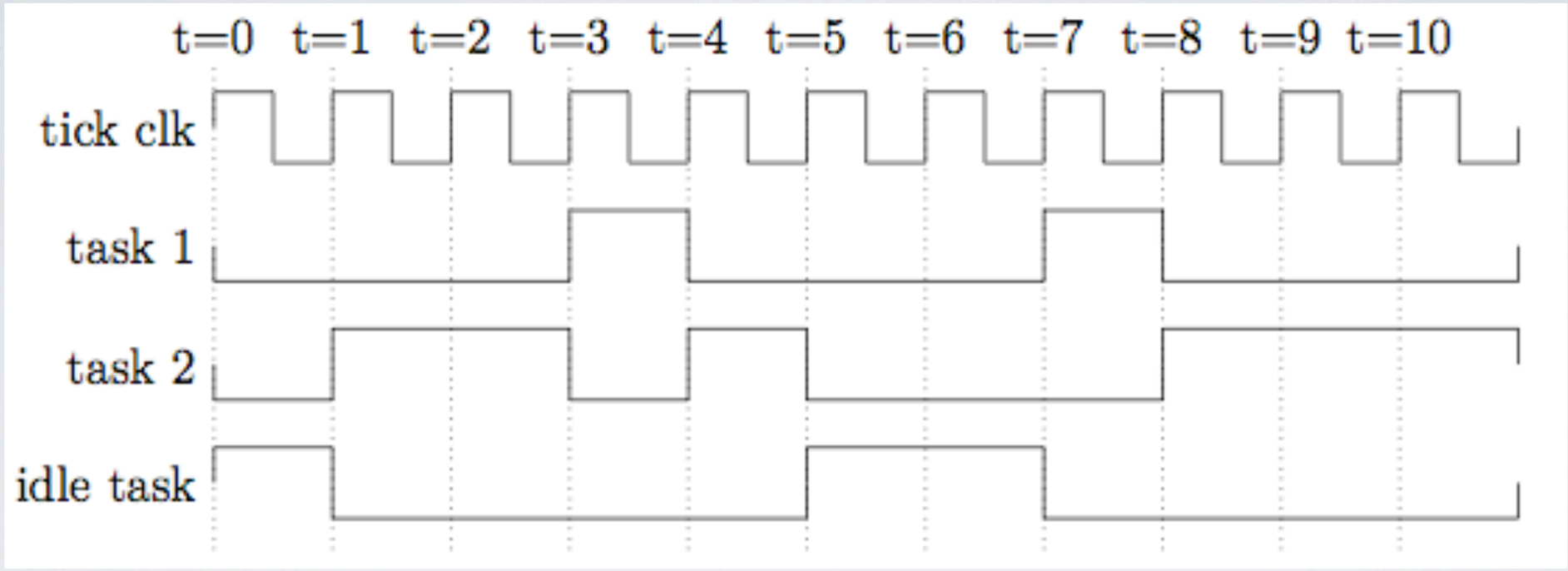
COOPERATIVE MULTITASKING

- Task give up access to the processor when done.

- The program implements in each task the giving up of the access to the processor.

- **Advantage**: OS is lighter, few RAM used
- **Disadvantage**: programmer must be conscientious, not precise timing

PREEMTIVE MULTITASKING



- Each task has a priority level, defined by the programmer
- Scheduler

- **Advantage**: abstraction for the programmer, precise timing
- **Disadvantage**: bigger OS, need more RAM

DEMONSTRATION APP.

- Task A, performs an analog to digital conversion to read the value of a photo-resistor.

- Task B, performs an analog to digital conversion to read the value of a thermistor.

- Task C, checks the state of an optical switch, if yes its sends an error message to the computer within 10 milliseconds.

- Task D, sends all information through USB to a computer.

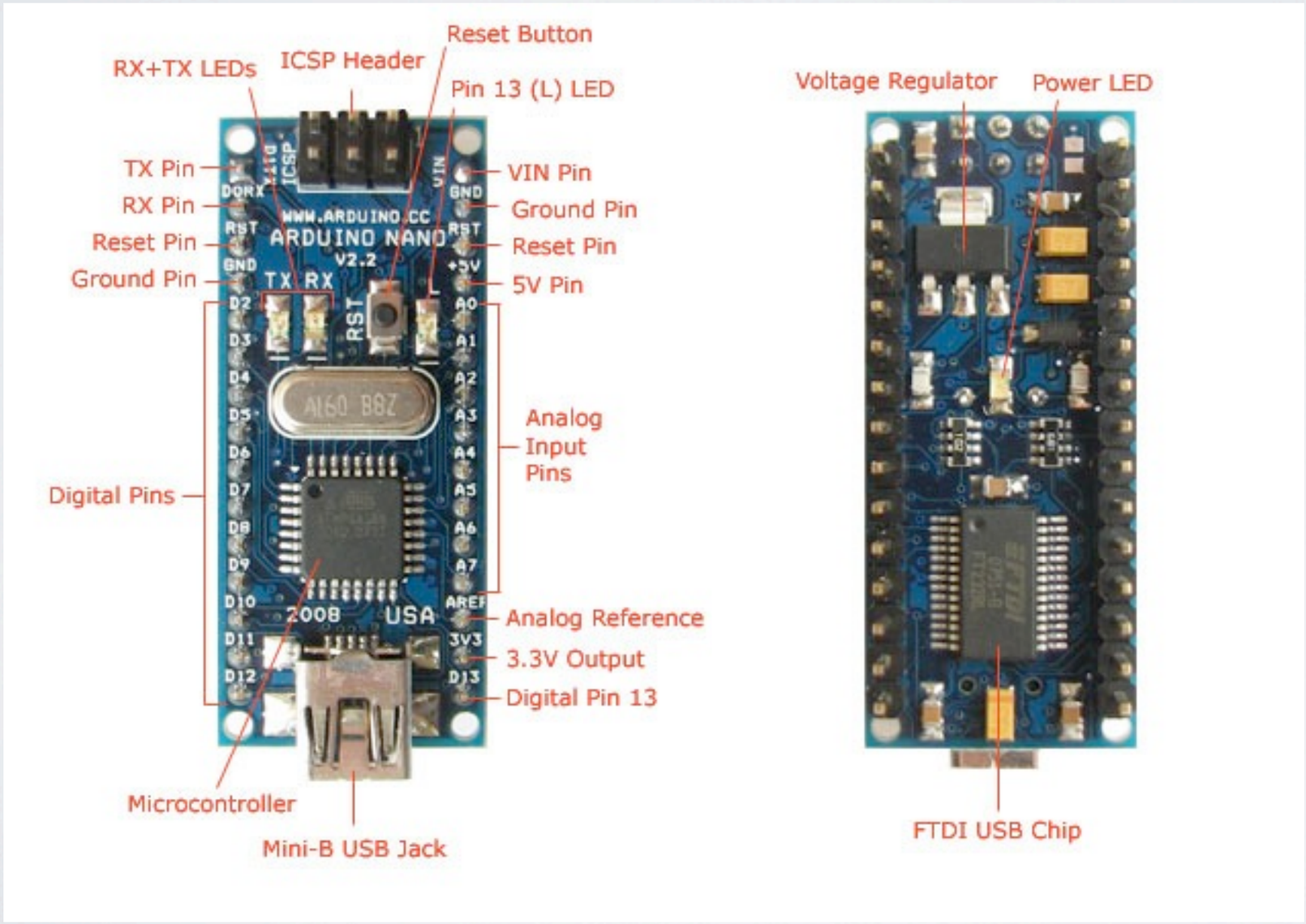
- Task E, checks if a message is received from the computer, if yes it launches a counter within 5 milliseconds.

DEMONSTRATION APP.

Task \ Property	Priority	Time constraint
A	2	Every 10ms
B	2	Every 10ms
C	3	Every 10ms
D	1	Every 1s
E	4	Every 5ms
Idle Task	1	-

ARDUINO

Open-source electronics prototyping platform
based on flexible easy-to-use hardware and software



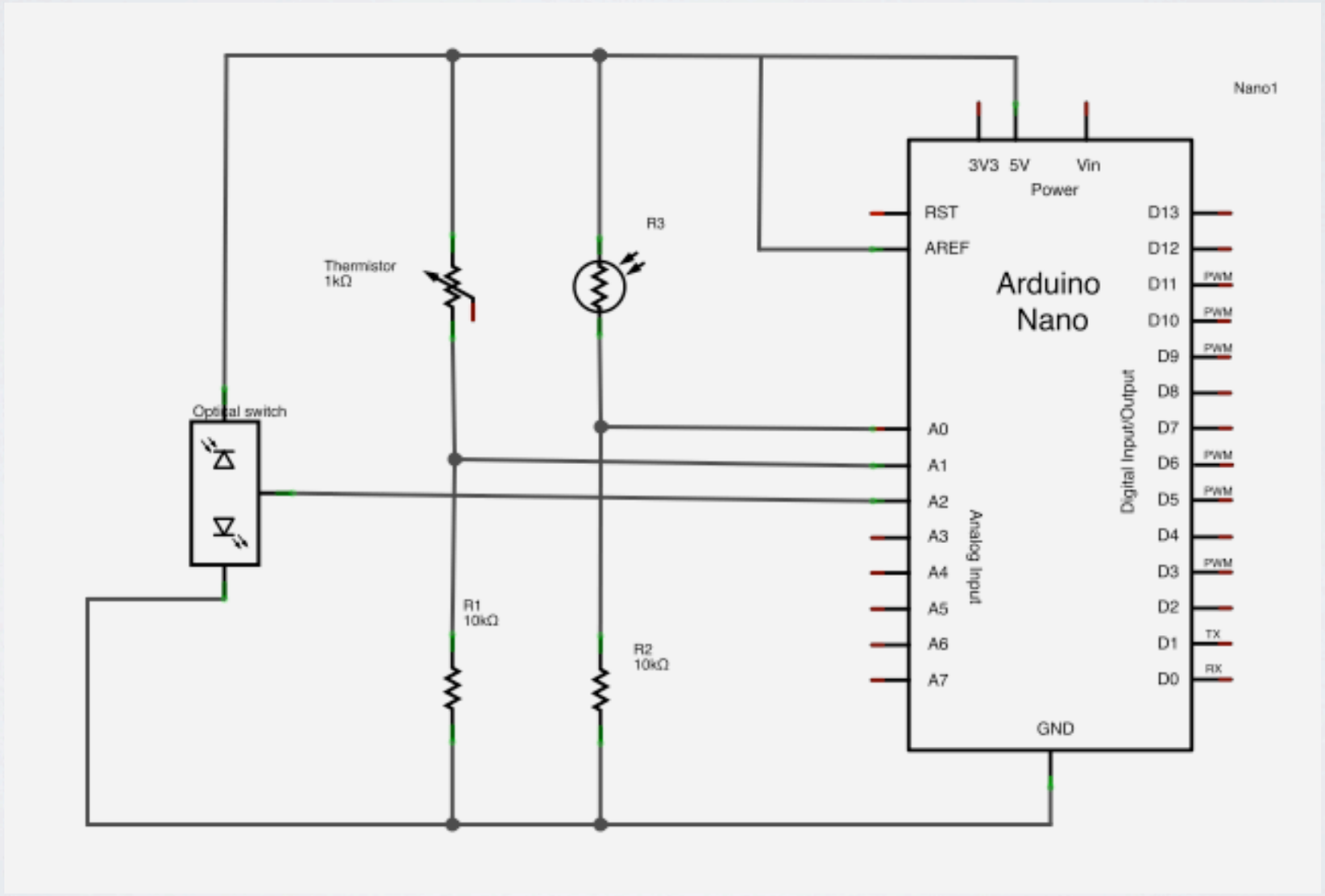
FREERTOS

Compatible with ATmega328 chipset.

- Compatible with the Arduino development board.
- Compatible with the last Arduino IDE (version 1.0).
- Prioritized preemptive scheduler
- Light enough to run correctly on an Arduino Nano.
- A good documentation
- Free

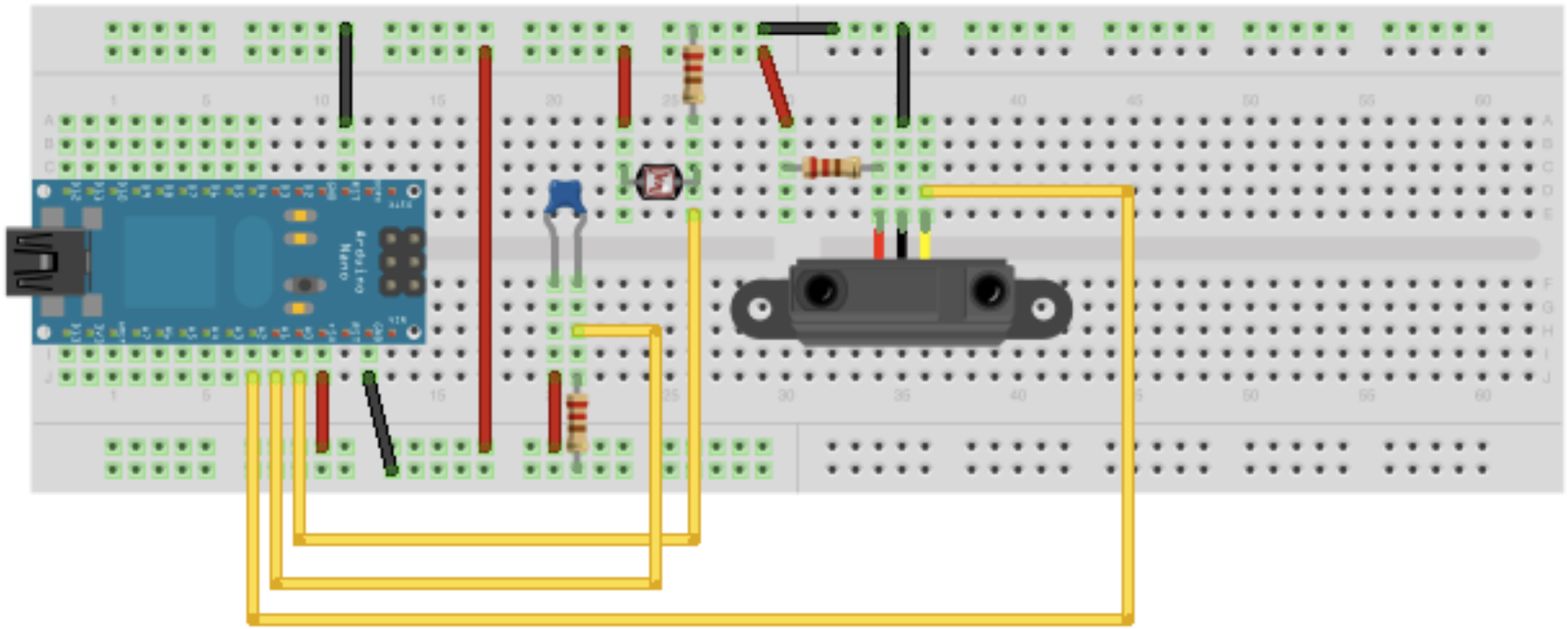
IMPLEMENTATION

Schematic of the system



IMPLEMENTATION

Protoboard



IMPLEMENTATION

How to implement a task with FreeRTOS ?

```
xTaskCreate(vTaskOne,
    (signed portCHAR *) "taskOne",
configMINIMAL STACK SIZE + 50,
NULL,
tskIDLE PRIORITY ,
NULL);
```

```
static void vTaskOne(void *pvParameters) {
    //Initialization code
    while (1) {
        //Task application code
    }
}
```

IMPLEMENTATION

Reading temperature sensor value

Mutex: token to access a global variable

```
static void vTemperatureTask(void *pvParameters)
{
	vTaskSetApplicationTaskTag(NULL, (char(*)(void*)) 2);
	while(1)
	{
		 if (xSemaphoreTake(xSemaphoreTemperature, portMAX_DELAY == pdTRUE))
		{
			temperature = analogRead(temperatureSensorPin);
			xSemaphoreGive(xSemaphoreTemperature);
		}
	vTaskDelay(configTICK_RATE_HZ/100);  //10ms
	}
}
```

IMPLEMENTATION

Testing optical switch

```
static void vOpticalSwitchTask(void *pvParameters)
{
    vTaskSetApplicationTaskTag(NULL,  (char(*)(void*))4);
    unsigned int temp;
    while(1)
    {
        if(analogRead(opticalSwitchSensorPin) > 50)
        {
            opticalSwitch = true;
            Serial.println("Error");
        }
        else opticalSwitch = false;
        vTaskDelay(configTICK_RATE_HZ/100); //10ms
    }
}
```

IMPLEMENTATION

Sending informations to the computer by serial

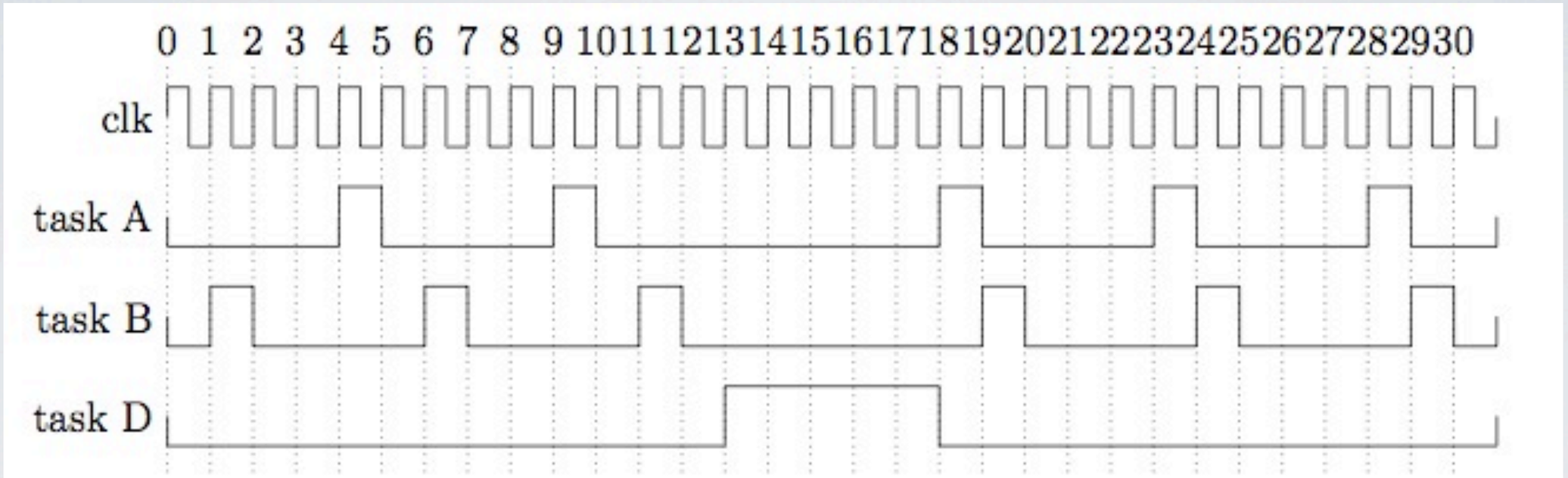
```
static void vSerialInfoTask(void *pvParameters) {
    vTaskSetApplicationTaskTag(NULL, (char(*)(void*))3);
    while(1) {
        if (xSemaphoreTake(xSemaphoreLuminosity, portMAX_DELAY == pdTRUE) and
        xSemaphoreTake(xSemaphoreTemperature, portMAX_DELAY == pdTRUE))
        {
            Serial.print("Luminosity: ");
            Serial.println(luminosity);
            Serial.print("Temperature: ");
            Serial.println(temperature);
            Serial.print("Optical switch: ");
            Serial.println(opticalSwitch);
            xSemaphoreGive(xSemaphoreLuminosity);
            xSemaphoreGive(xSemaphoreTemperature);
        }
        vTaskDelay(configTICK_RATE_HZ); // 1 secondes waiting
    }
}
```

IMPLEMENTATION

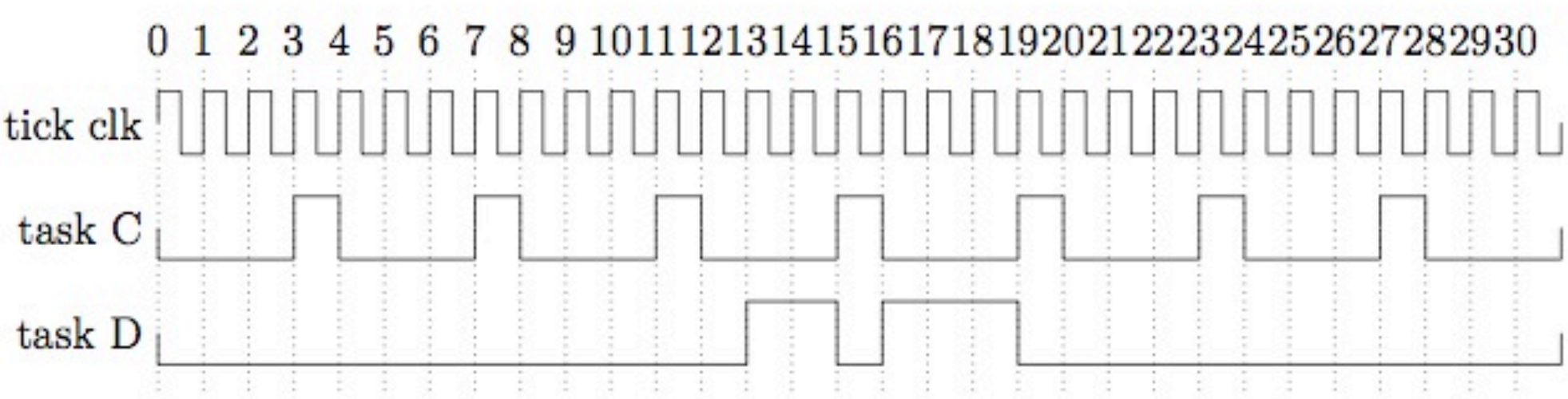
Retreiving informations from the computer by serial

```
static void vSerialMsgCheck(void *pvParameters)
{
    vTaskSetApplicationTaskTag(NULL, (char(*)(void*))5);
    while(1) {
        if (Serial.available() > 0)
        {
          int receivedByte = Serial.read();
                if (receivedByte == 's')
                {
                        delay(3000); //Force waiting 1 second
                }
        }
        vTaskDelay(configTICK_RATE_HZ/200); //5ms
    }
}
```

RESULTS

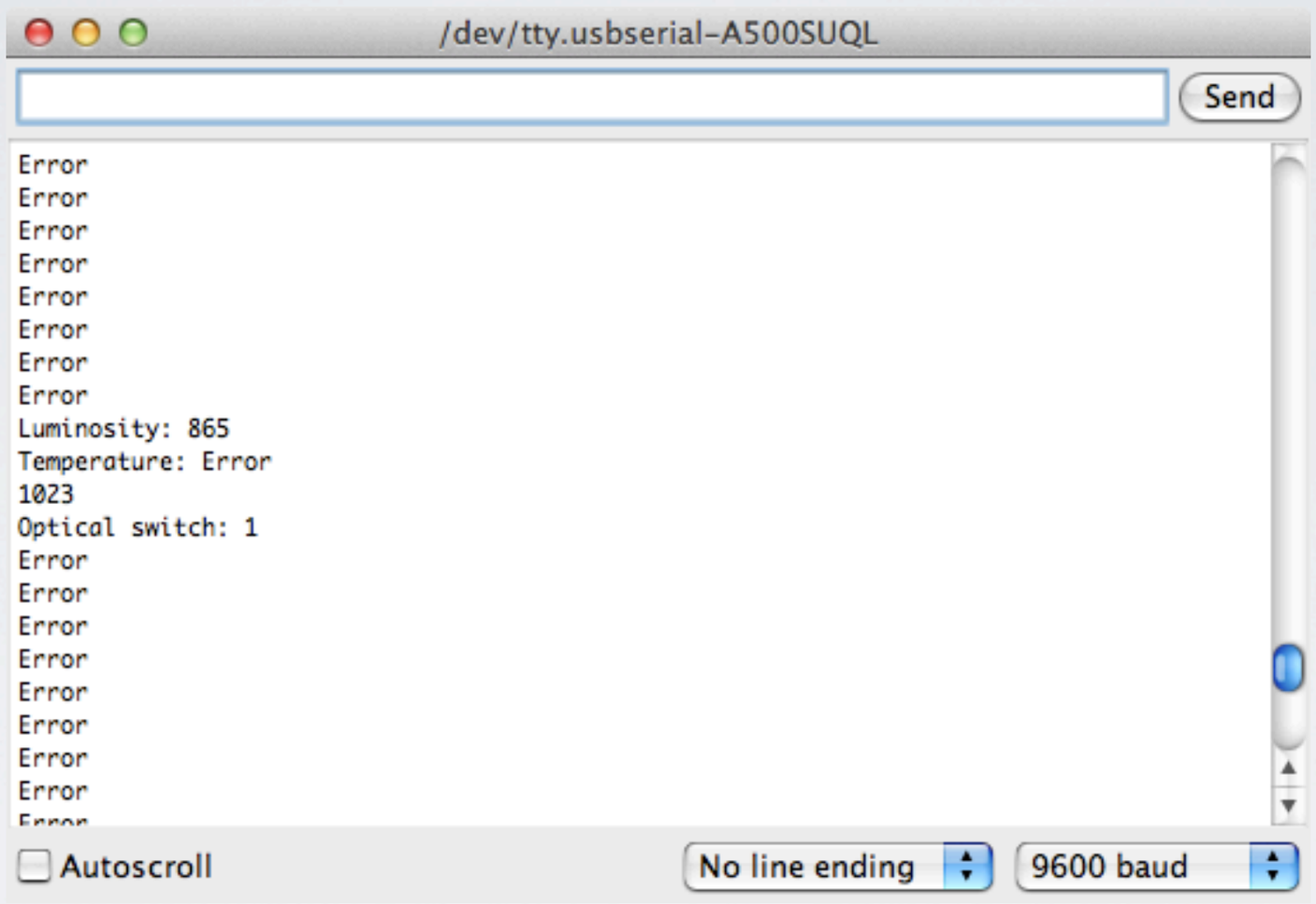


Effect of the use of mutex's



Scheduler pausing lower priority task D to allows task C to run.

RESULTS


/dev/tty.usbserial-A500SUQL
Send
Error
Error
Error
Error
Error
Error
Error
Error
Luminosity: 865
Temperature: Error
1023
Optical switch: 1
Error
Error
Error
Error
Error
Error
Error
Error
Error
Autoscroll
No line ending
9600 baud