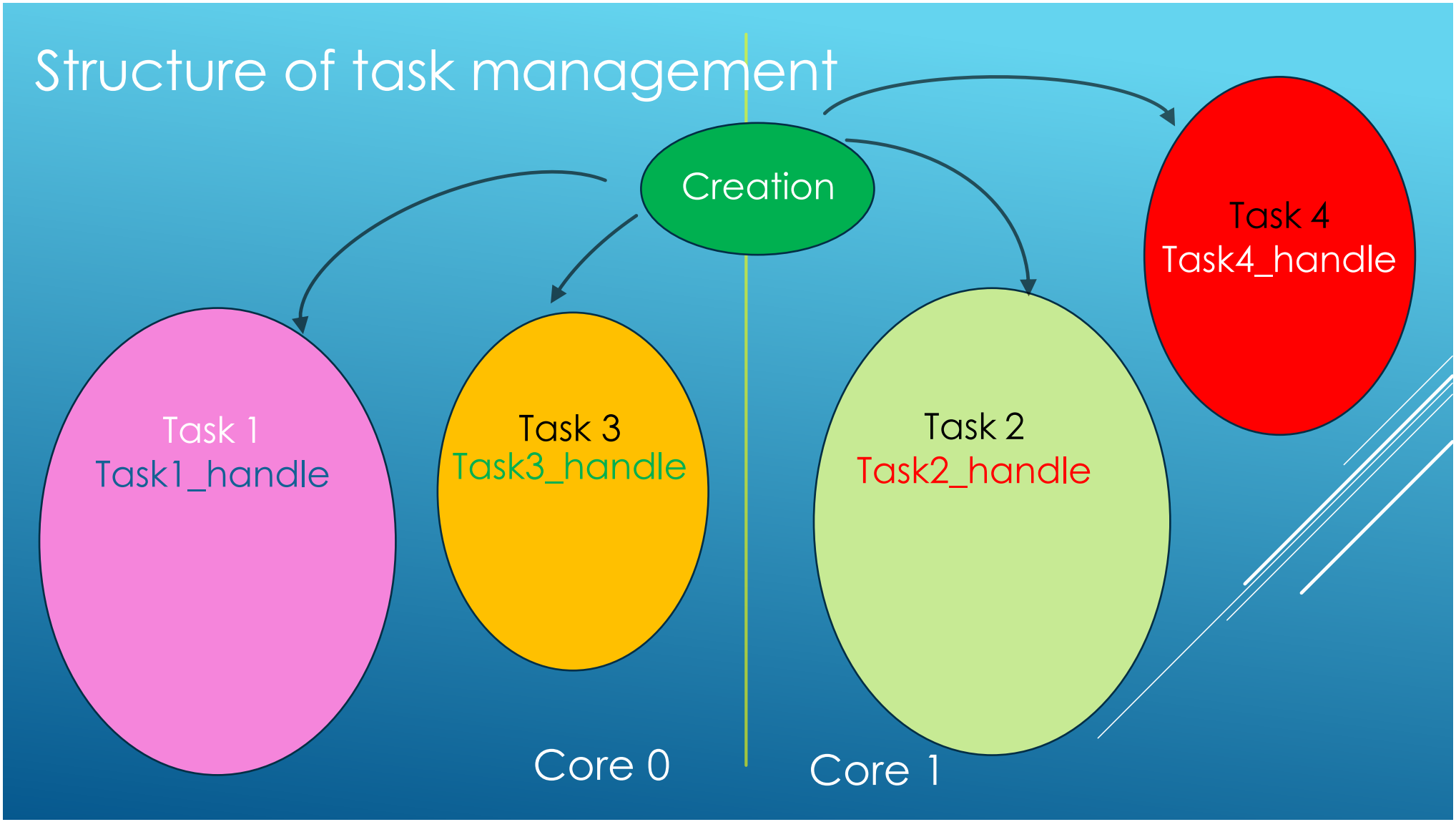
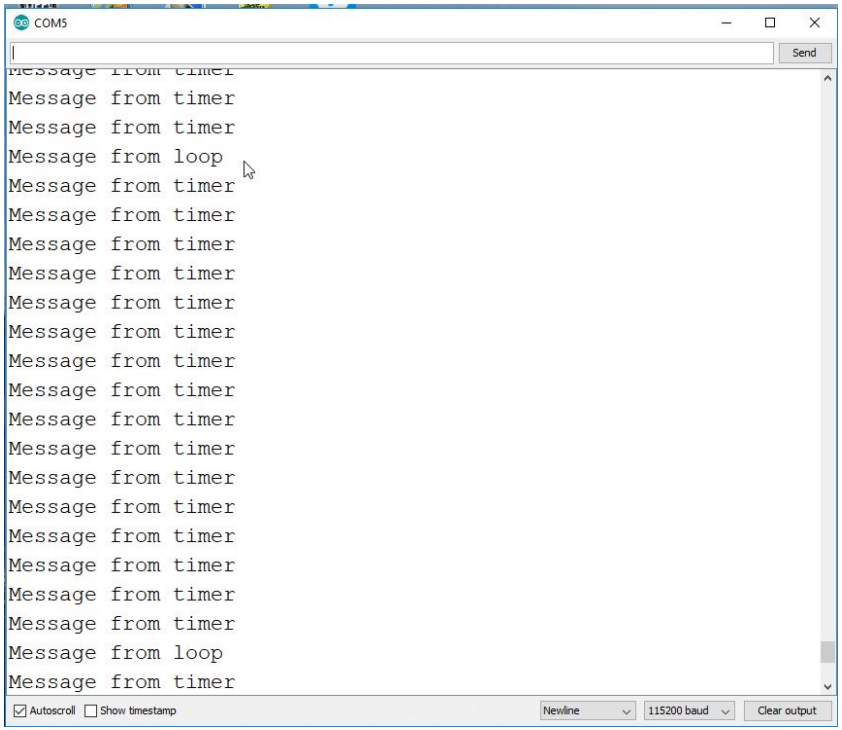
สำหรับการฝึกอบรมเชิงปฏิบัติการ "อุปกรณ์ไอโอทีสำหรับงานควบคุมอุตสาหกรรม"
ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ ม.นเรศวร 26-27 มิถุนายน 2564

FreeRTOS Basics

Dr.Varodom Toochinda
Dept. of Mechanical Engineering,
Kasetsart University

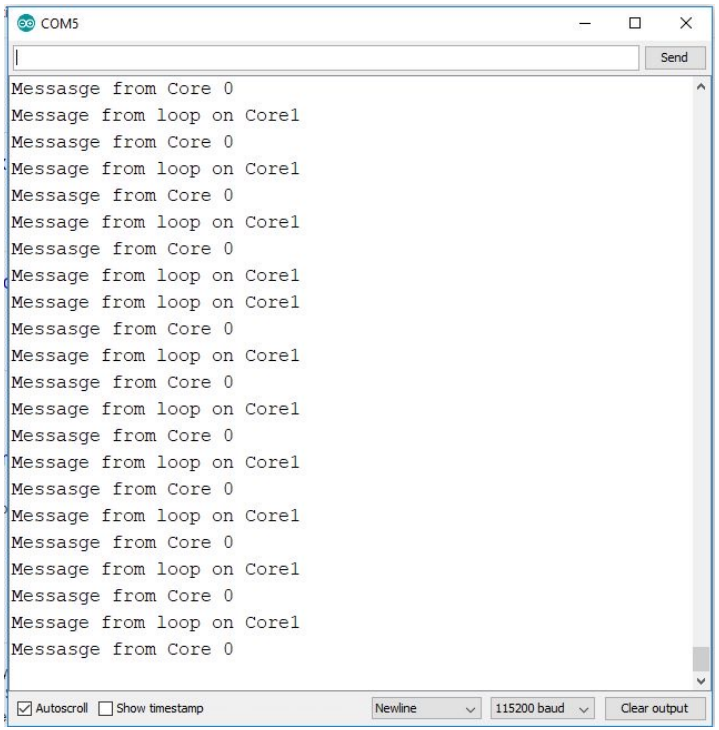
from chapter 8 of "IoT Development Handbook"


Structure of task management
Creation
Task 4
Task4_handle
Task 1
Task1_handle
Task 3
Task3_handle
Task 2
Task2_handle
Core 0
Core 1



Lab2_0a.ino : timer v.s. loop

both executed on core 1



Lab2_0b.ino : core0 task v.s. loop

use xPortGetCoreID() to check
on which core the task is running)

Simple Task creation (without specifying a core)

```
void myTask1( void * parameter );
xTaskCreate(    myTask1,           /* Task function. */
                "TaskOne",         /* String with name of task. */
                10000,             /* Stack size in words. */
                NULL,          /* Parameter passed as input of the task */
                 1,                /* Priority of the task. */
                Task1h);             /* Task handle. */
```

Task created with xTaskCreate() is executed on core 0

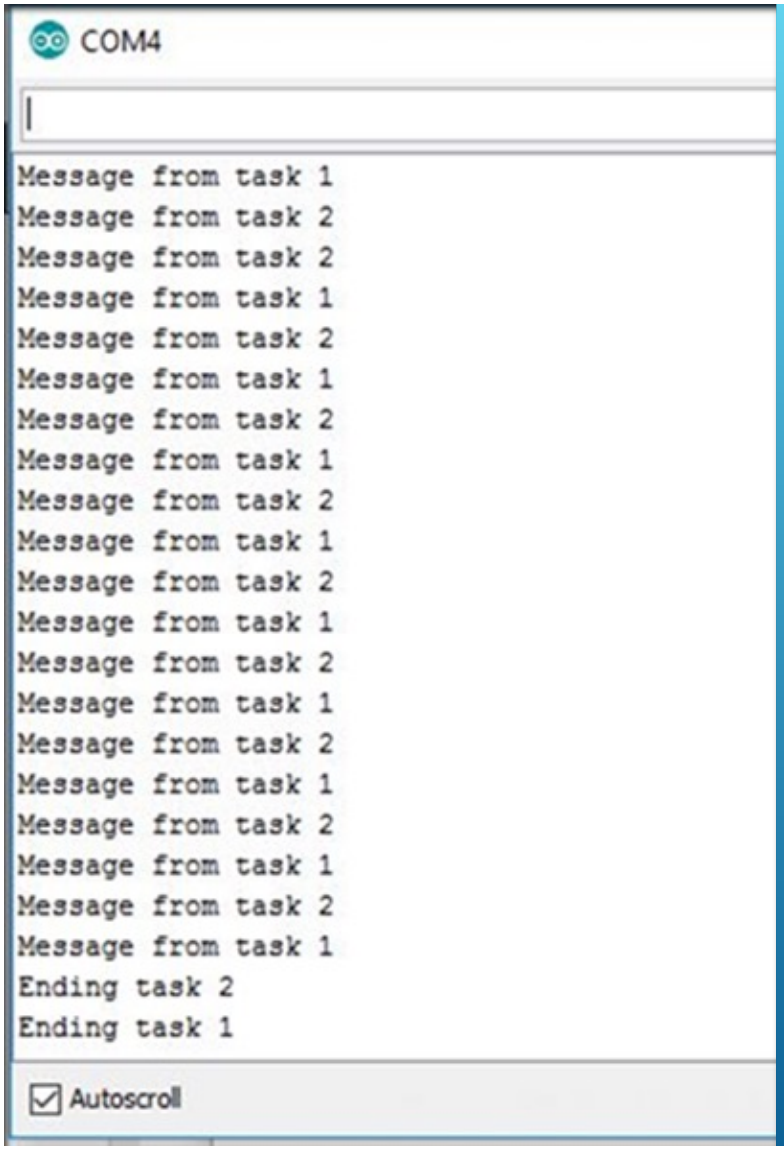
ex2_1.ino : task creation with equal priority

```
xTaskCreate(myTask1,  "TaskOne",  10000, NULL, 1,  NULL);
xTaskCreate(myTask2,  "TaskTwo",  10000, NULL, 1,  NULL);
```

```
void myTask1( void * parameter )
{
    volatile uint32_t ul;
    for( int i = 0;i<10;i++ ){
        Serial.println("Message from task 1");
        for (ul=0;ul<2000000;ul++);
}
    Serial.println("Ending task 1");
    vTaskDelete( NULL );
 }
```

```
void myTask2( void * parameter)
{
    volatile uint32_t ul;
    for( int i = 0;i<10;i++ ){

        Serial.println("Message from task 2");
        for (ul=0;ul<2000000;ul++);
}
    Serial.println("Ending task 2");
    vTaskDelete( NULL );
 }
```



ex2_1.ino
output on Serial monitor

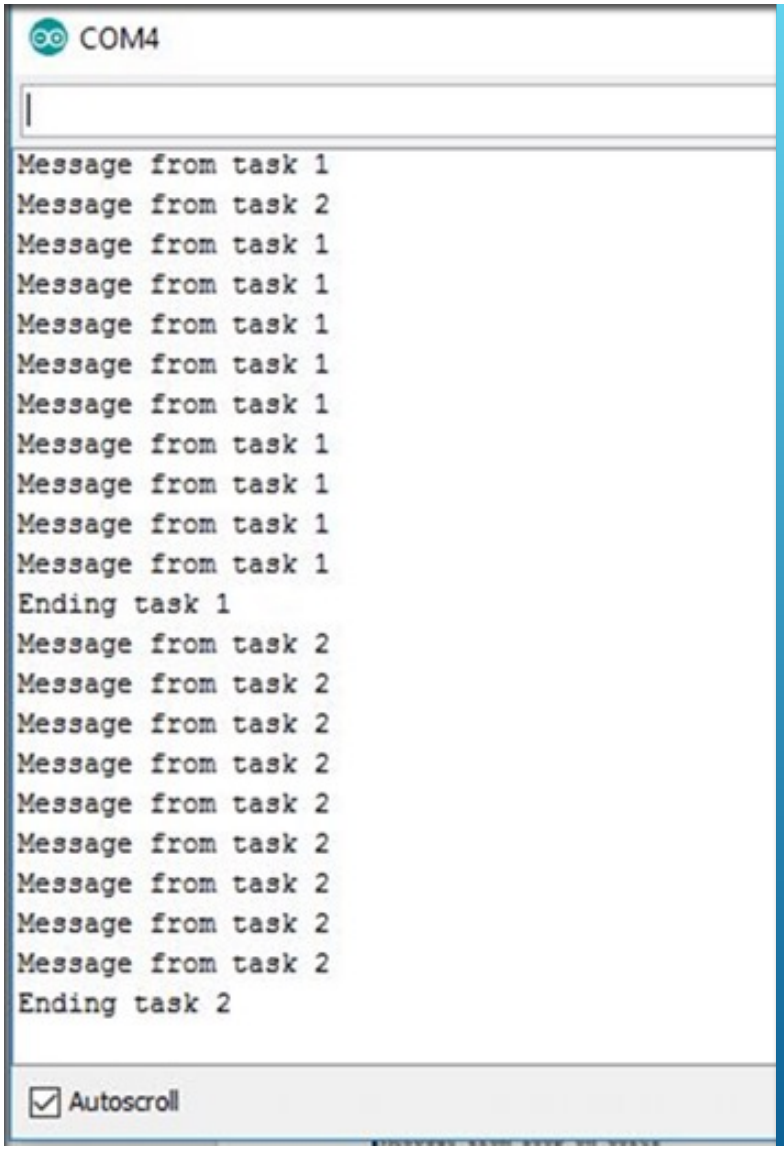
ex2_2.ino task creation with specified priority

```
xTaskCreate(myTask1,  "TaskOne",  10000, NULL, 1,  NULL);
xTaskCreate(myTask2,  "TaskTwo",  10000, NULL, 0,  NULL);
```

```
void myTask1( void * parameter )
{
    volatile uint32_t ul;
    for( int i = 0;i<10;i++ ){
        Serial.println("Message from task 1");
        for (ul=0;ul<2000000;ul++);
}
    Serial.println("Ending task 1");
    vTaskDelete( NULL );
 }
```

```
void myTask2( void * parameter)
{
    volatile uint32_t ul;
    for( int i = 0;i<10;i++ ){

        Serial.println("Message from task 2");
        for (ul=0;ul<2000000;ul++);
}
    Serial.println("Ending task 2");
    vTaskDelete( NULL );
 }
```

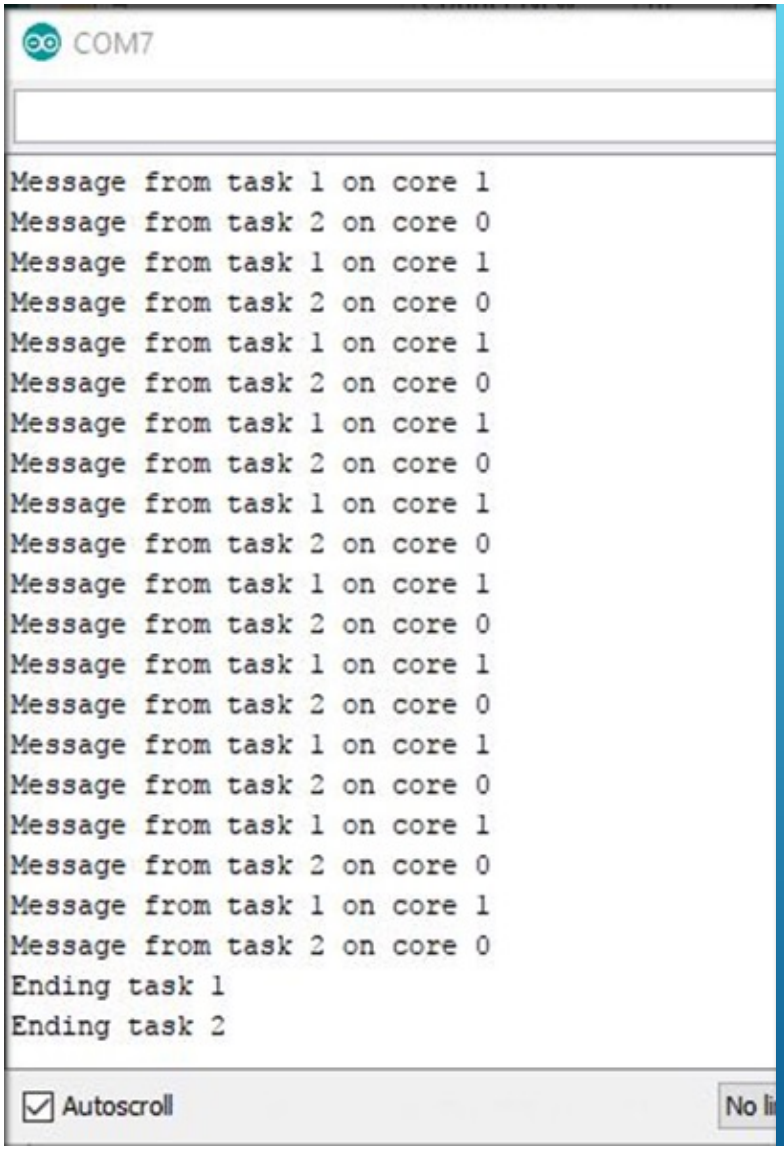


ex2_2.ino output on Serial monitor

ex2_3.ino task creation with specified core

```
xTaskCreatePinnedToCore(
            myTask1,         /* Task function. */
            "TaskOne",       /* String with name of task. */
            10000,           /* Stack size in words. */
            NULL,            /* Parameter input of the task */
            1,               /* Priority of the task. */
            NULL,            /* Task handle. */
            1);              /* assign Task 1 to Core 1 */
```

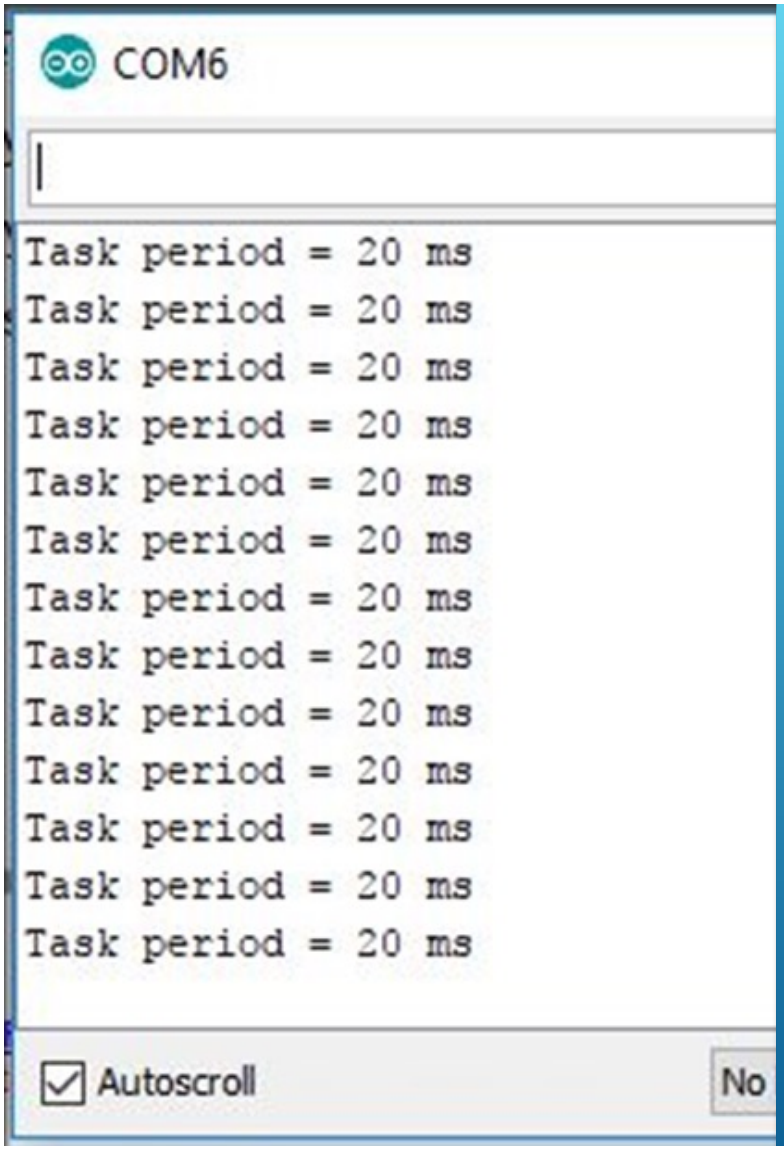
```
xTaskCreatePinnedToCore(
            myTask1,         /* Task function. */
            "TaskOne",       /* String with name of task. */
            10000,           /* Stack size in words. */
            NULL,            /* Parameter input of the task */
            0,               /* Priority of the task. */
            NULL,            /* Task handle. */
            0);              /* assign Task 2 to Core 0 */
```



ex2_3.ino output on serial monitor

ex2_4.ino : periodic task creation

```
void myPeriodicTask( void * parameter)
{
  TickType_t xLastWakeTime;
  xLastWakeTime = xTaskGetTickCount();
  for(;;)
  {
     told = tnew;
     tnew = millis();
     dt = tnew - told;
     vTaskDelayUntil(&xLastWakeTime, pdMS_TO_TICKS(20));
   // set period = 20 millisecs
  }
}
```

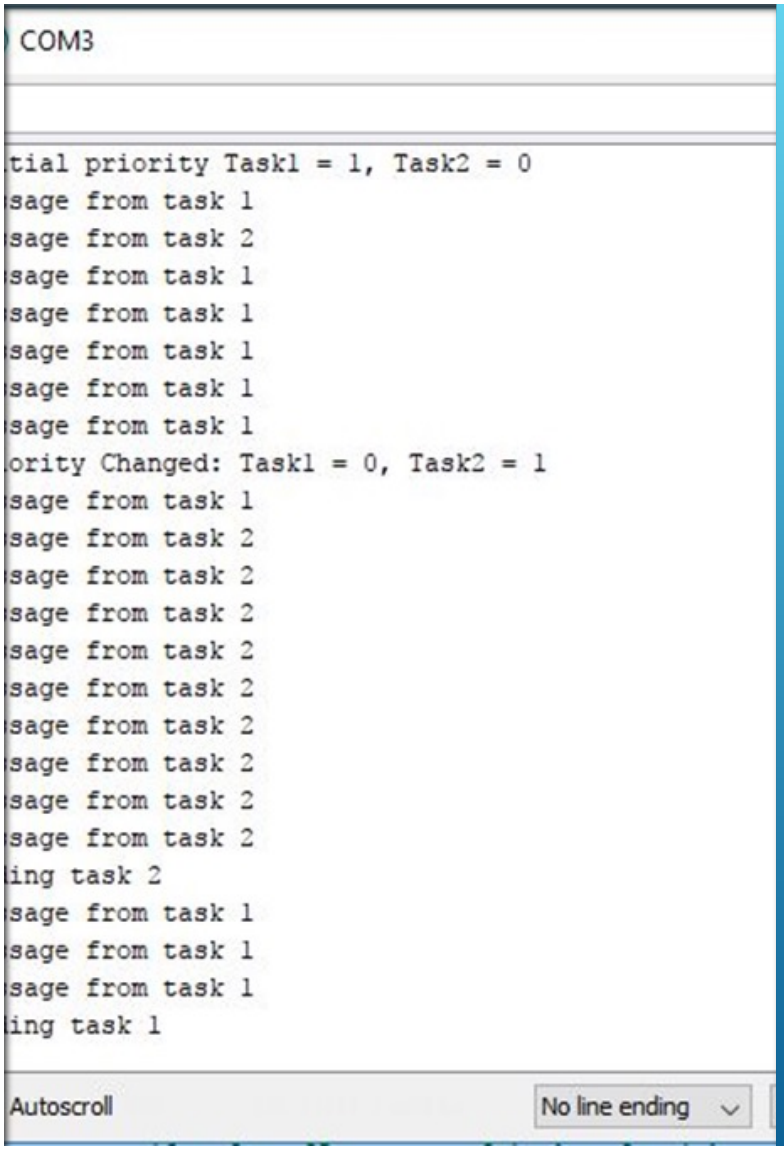


ex2_4.ino output on serial monitor

ex2_5.ino : change task priority after creation

add this code on Task1

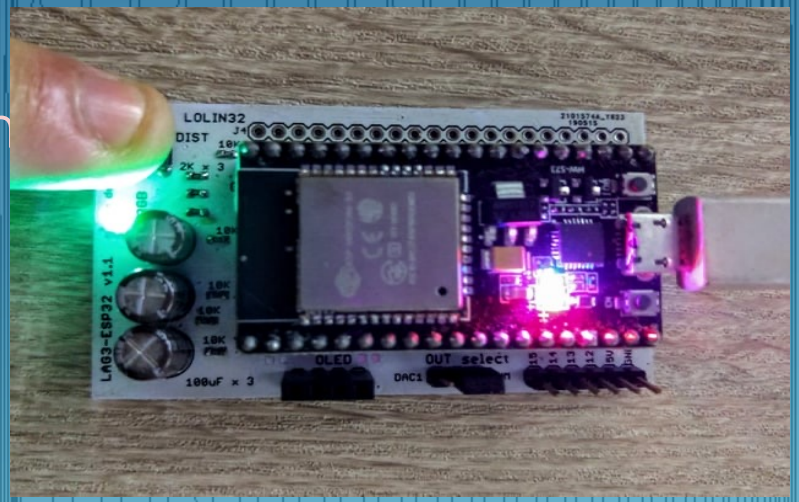
```
if (i==5){
   Serial.println("Priority Changed: Task1=0, Task2=1");
   vTaskPrioritySet(NULL, 0);  // decrease task 1 priority
   vTaskPrioritySet(xTask2Handle, 1); // increase task 2 priority
}
```



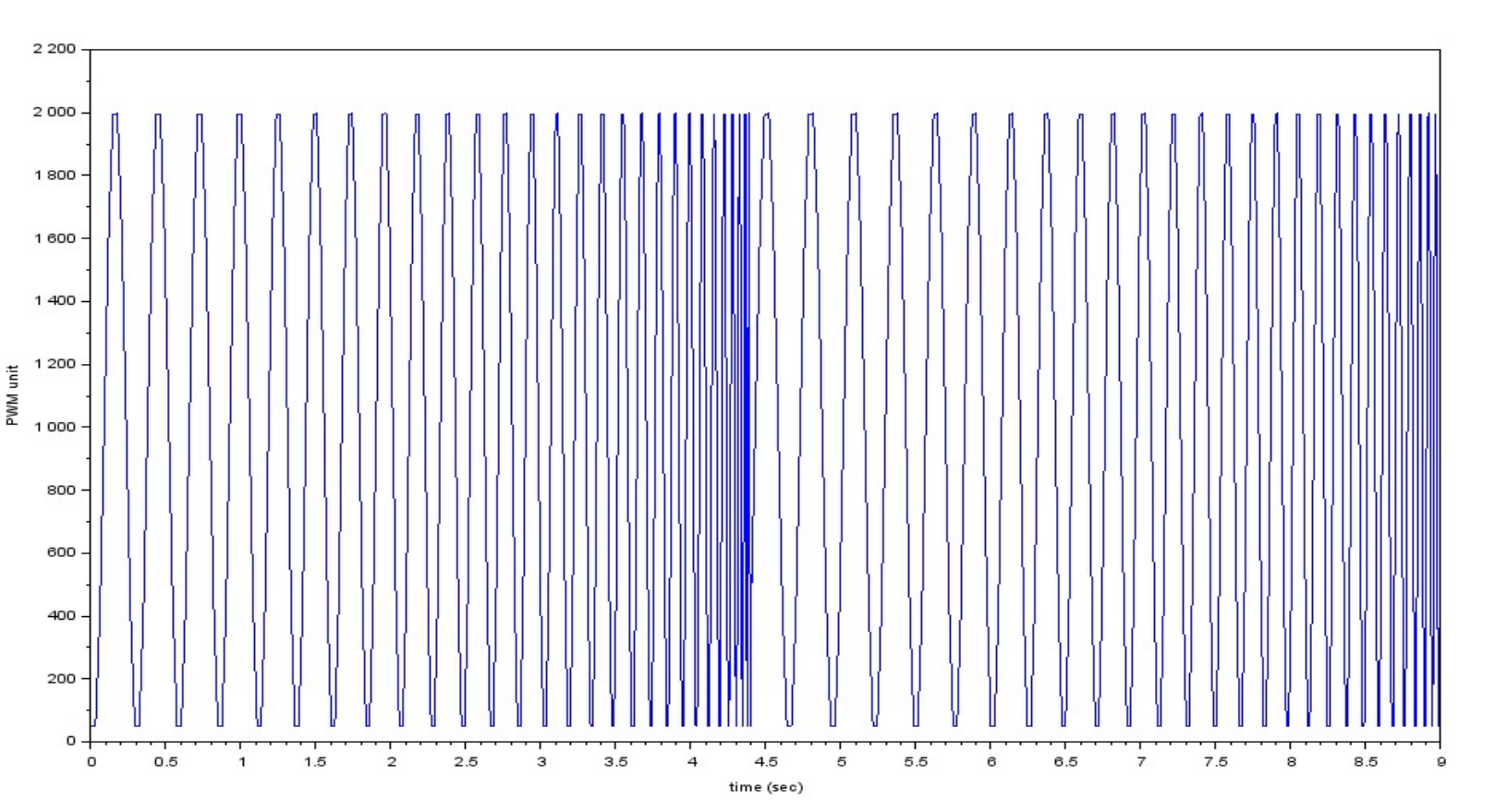
ex2_5.ino output on serial monitor

TRUE MULTI-TASKING

- Task 1 : blink LED
- Task 2 : generate chirp when switch is pressed
- Compare between
  - Multitask_basic.ino
  - Multitask_freeRTOS.ino



Chirp Signal generated By genchirp() function



Exercise 1 :

- Add a 1-second periodic task to Multitask_freeRTOS.ino that prints the period to serial monitor.
- This task should not stop when the switch is pressed.

Suspend/resume task

Use the following code

```
void vTaskSuspend( TaskHandle_t xTaskToSuspend );
void vTaskResume( TaskHandle_t xTaskToResume );
```

where the argument is the handle of task to be suspended/resume, or NULL if referred to the task that calls the function.

To suspend/resume all tasks, use

```
void vTaskSuspendAll( void );
BaseType_t xTaskResumeAll( void );
```

ex2_6.ino : queue creation

```
queue = xQueueCreate( 10, sizeof( int ) );
  if(queue == NULL){
    Serial.println("Error creating the queue");
  }
```

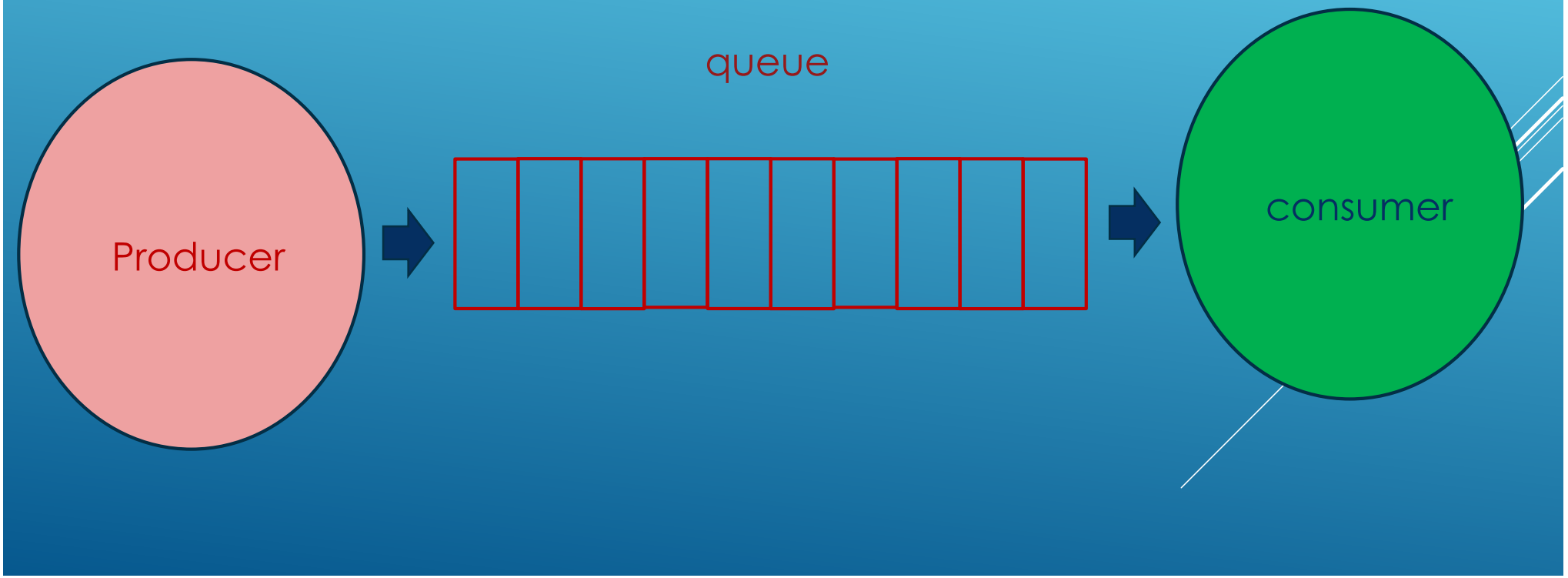
To write data to queue, use `xQueueSend()` with following format

```
xQueueSend(queue, &data, xTicksToWait);
```

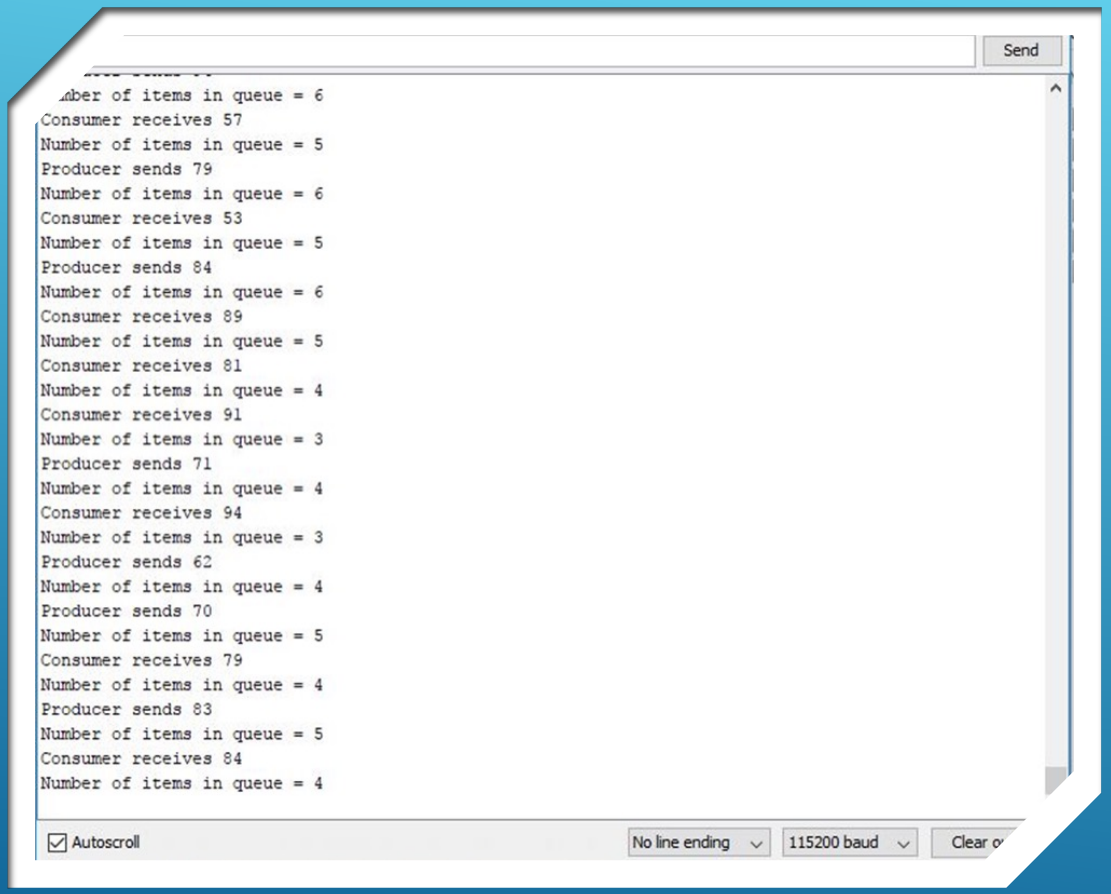
To read from queue, use

```
xQueueReceive(queue, &data, xTicksToWait);
```

ex2_6.ino : using queue

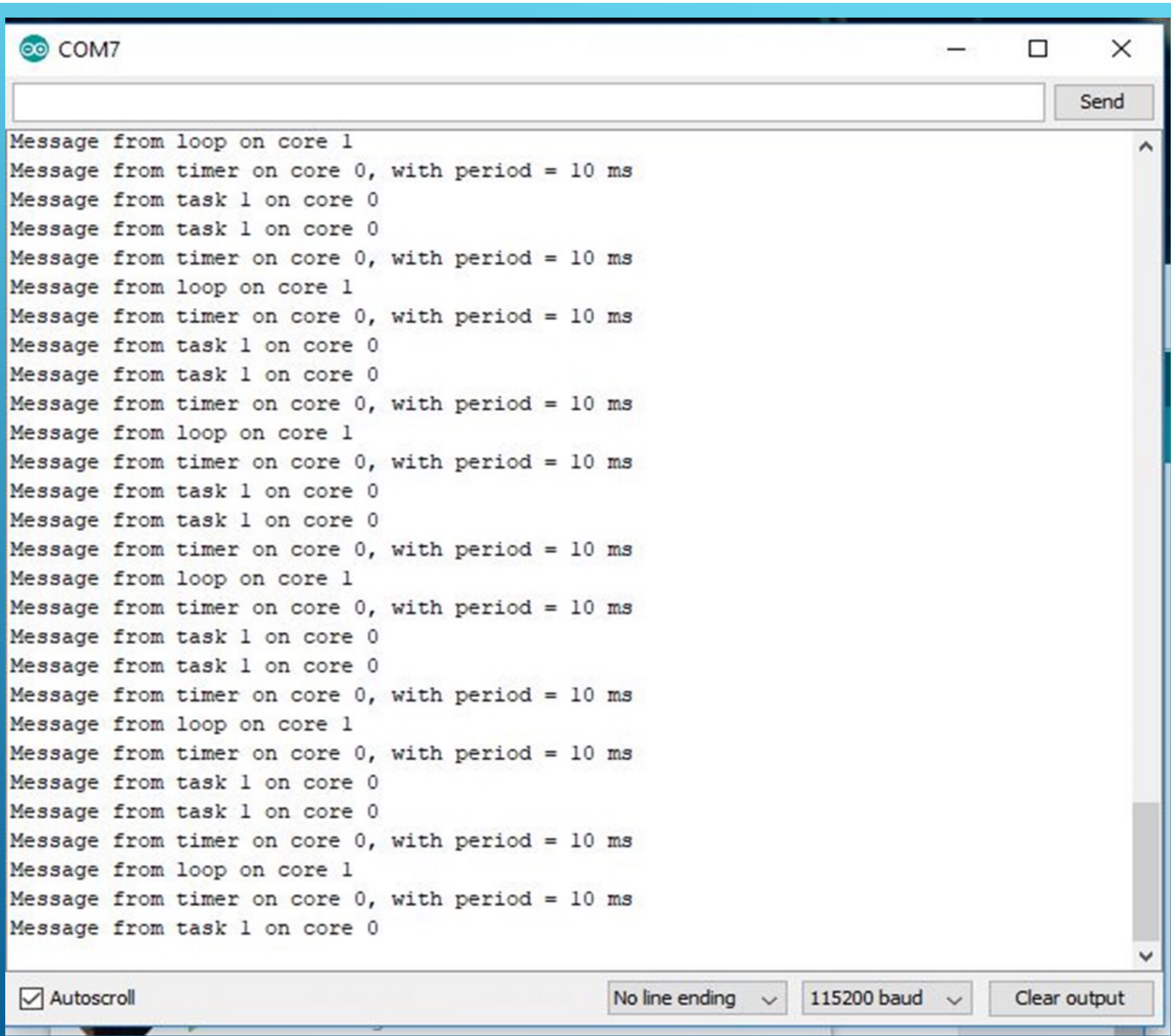


ex2_6.ino output on serial monitor



"No different from periodic task"

ex2_7.ino : timer task



ex2_7.ino output on serial monitor

Critical section protection

- use macro taskENTER_CRITICAL() and taskEXIT_CRITICAL()
- stop scheduler or tasks that access protected variables
- use semaphore or mutex

ex2_8.ino : deadlock problem

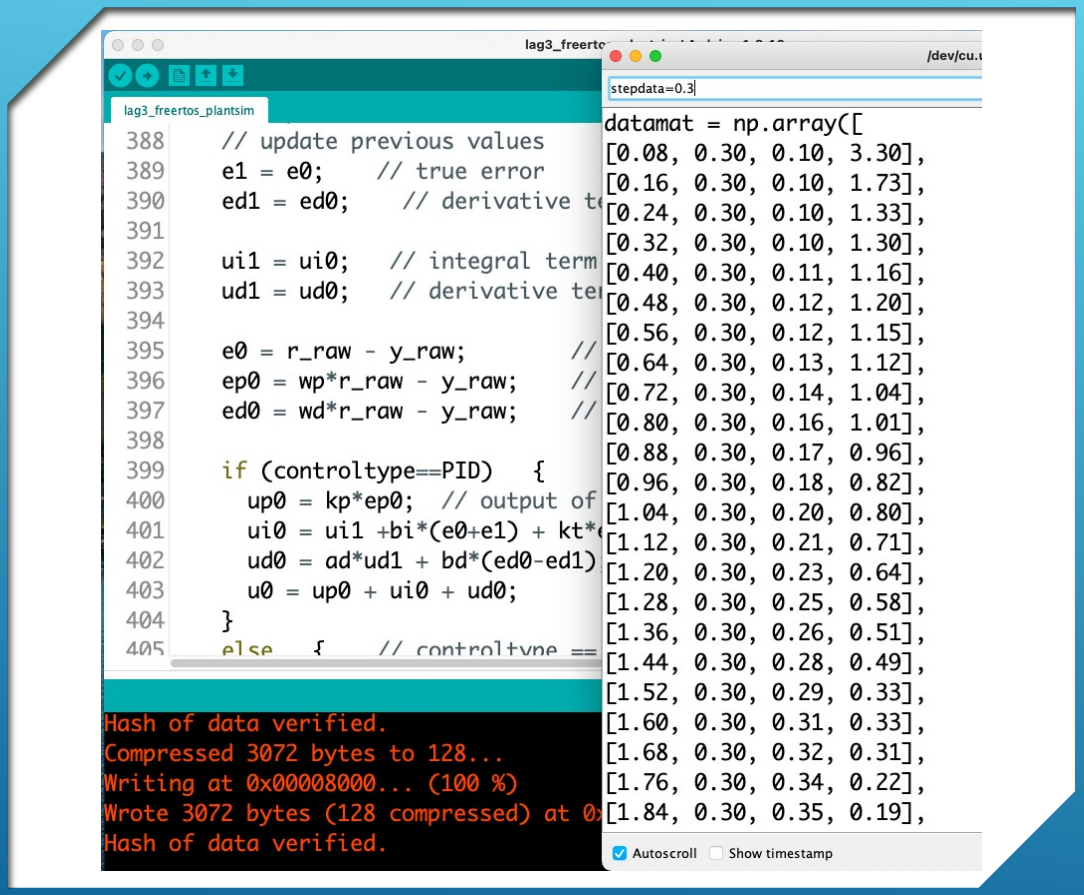
```
void Task1( void * parameter )
{
  for(;;){
    xSemaphoreTake( xMutex1, portMAX_DELAY );
    Serial.println("Task 1 has mutex 1");
    Serial.println("Task 1 now attempts to get mutex 2");
    xSemaphoreTake(xMutex2, portMAX_DELAY);
    delay(1000);
    xSemaphoreGive( xMutex1 );
    Serial.println("Task 1 releases mutex 1");
  }
  vTaskDelete( NULL );
}
```

```
void Task2( void * parameter )
{
  for(;;){
    xSemaphoreTake( xMutex2, portMAX_DELAY );
    Serial.println("Task 2 has mutex 2");
    Serial.println("Task 2 now attempts to get mutex 1");
    xSemaphoreTake(xMutex1, portMAX_DELAY);
    delay(1000);
    xSemaphoreGive( xMutex2);
  }
  vTaskDelete( NULL );
}
```

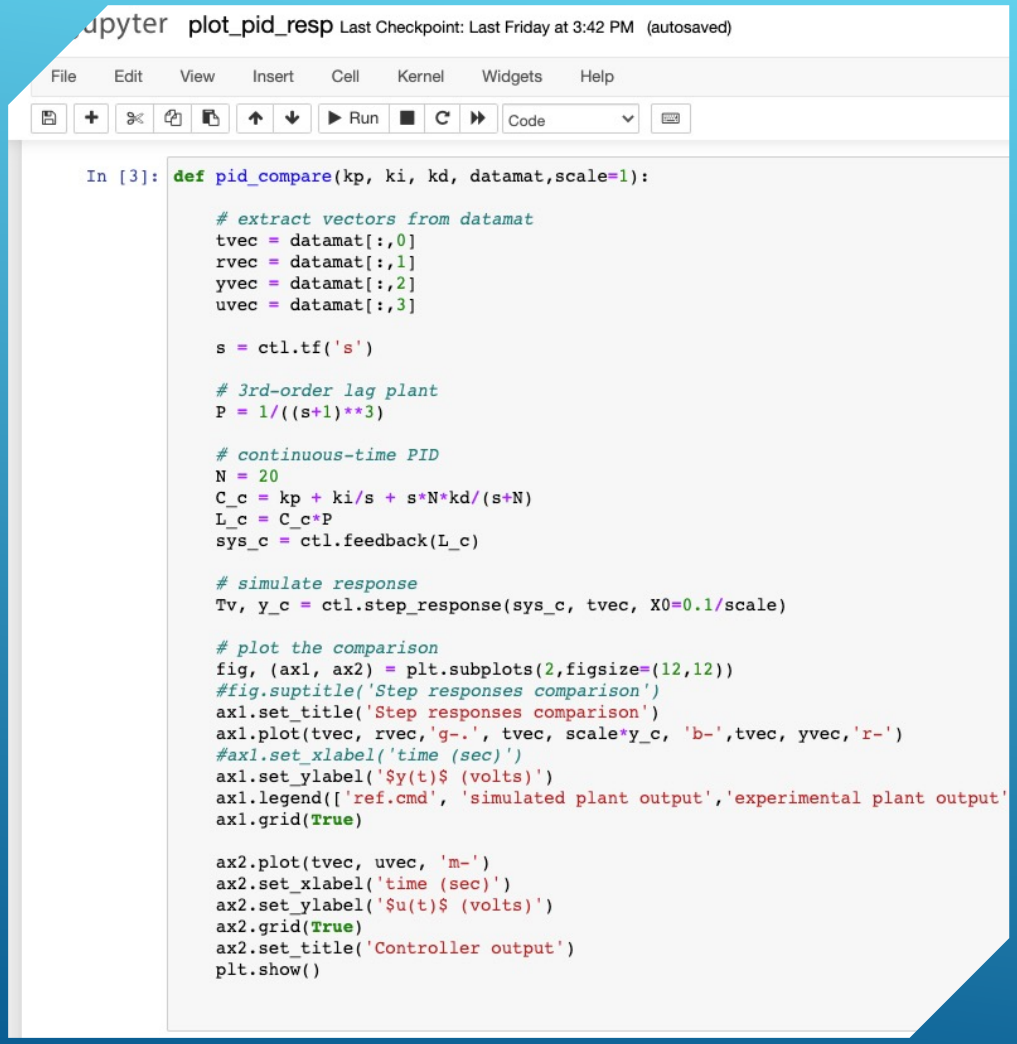
ex2_8.ino output on serial monitor

Task	Core	Function
0	0	Controller
1	1	Receive data from queue and send to serial port
3	1	RGB LED control
4	1	Command intepreter
5	1	NETPIE communication
7	1	Plant simulation

lag3_freertos_plantsim.ino

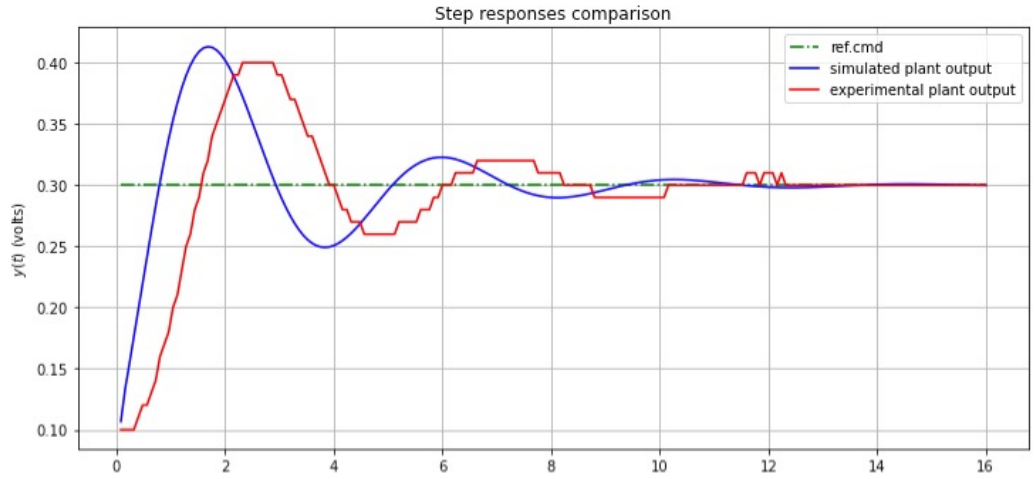


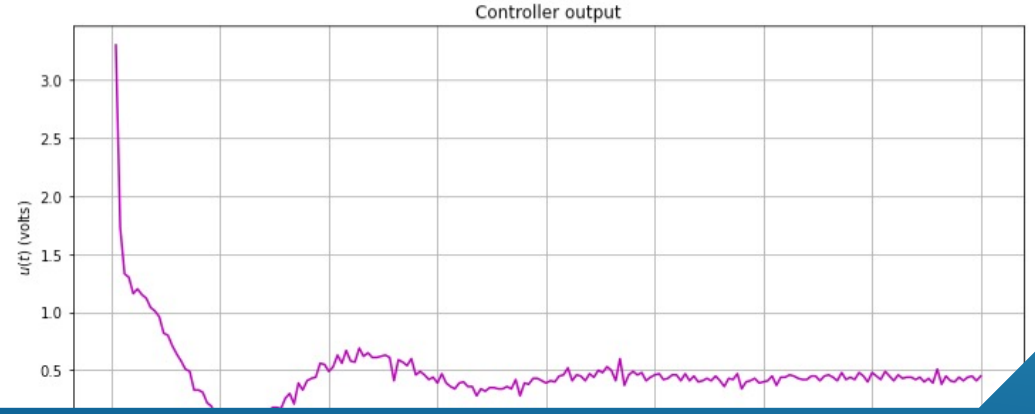
step data on serial monitor



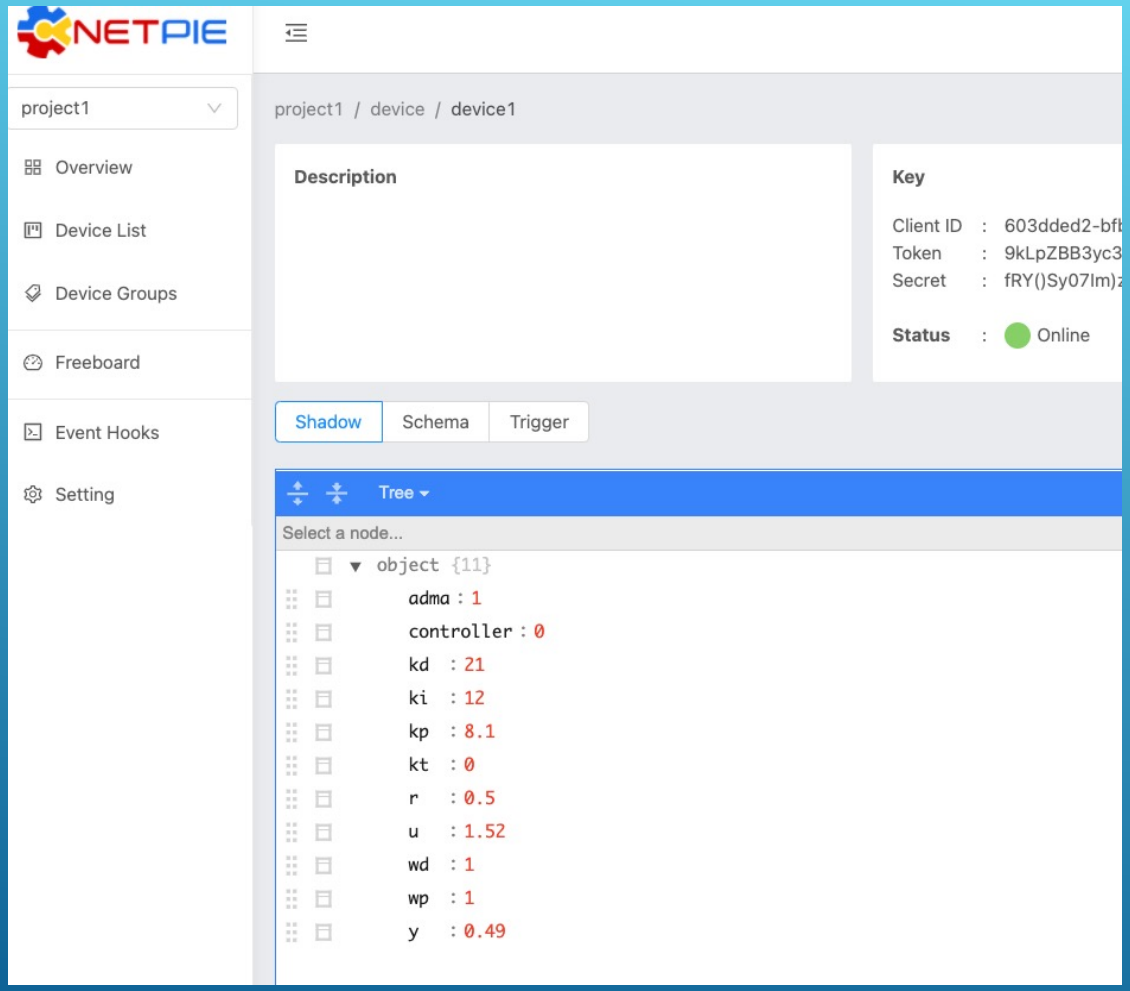
write Python function to compare step responses

```
pid_compare(kp,ki,kd,datamat,scale=0.3)
```

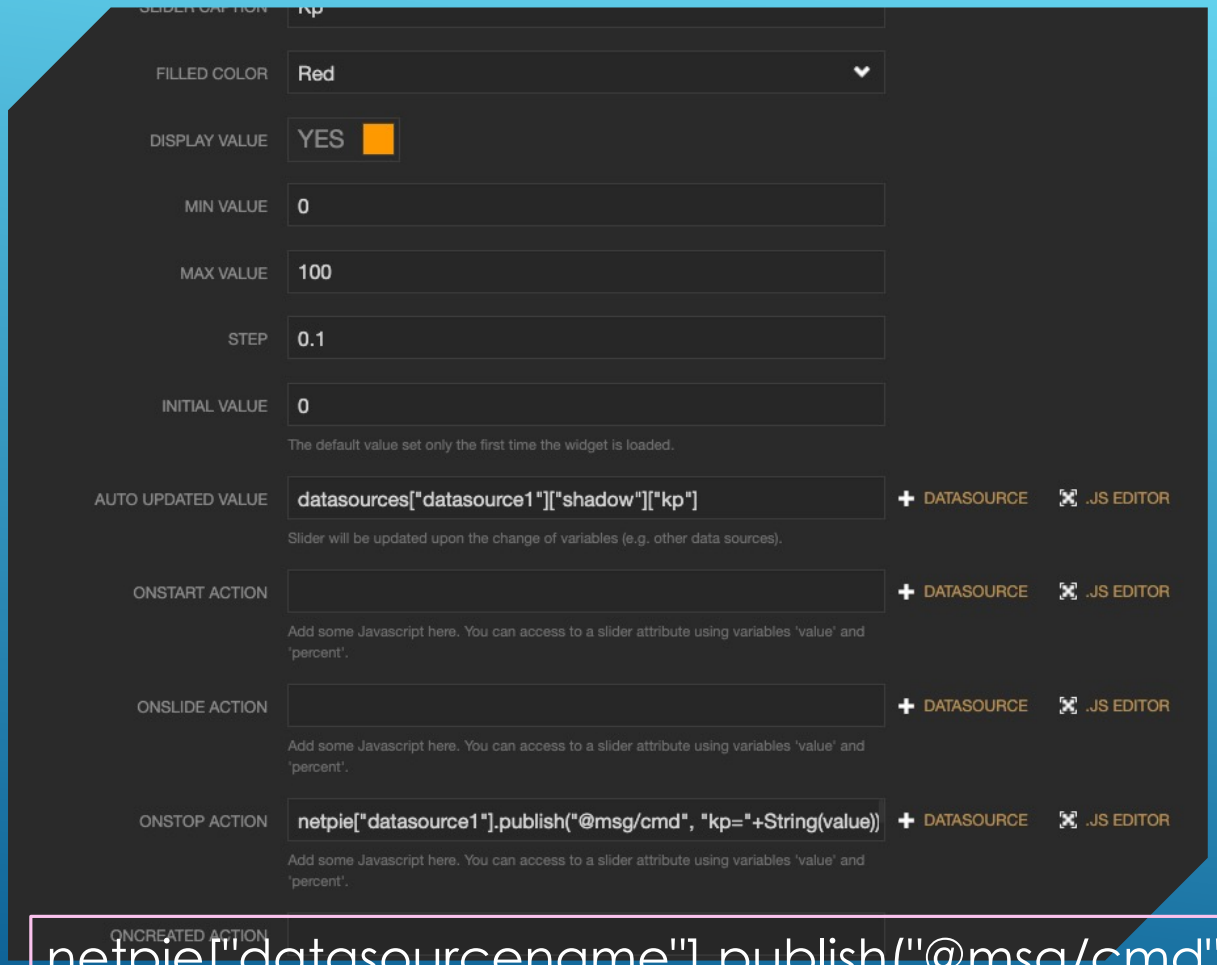




step response compare result



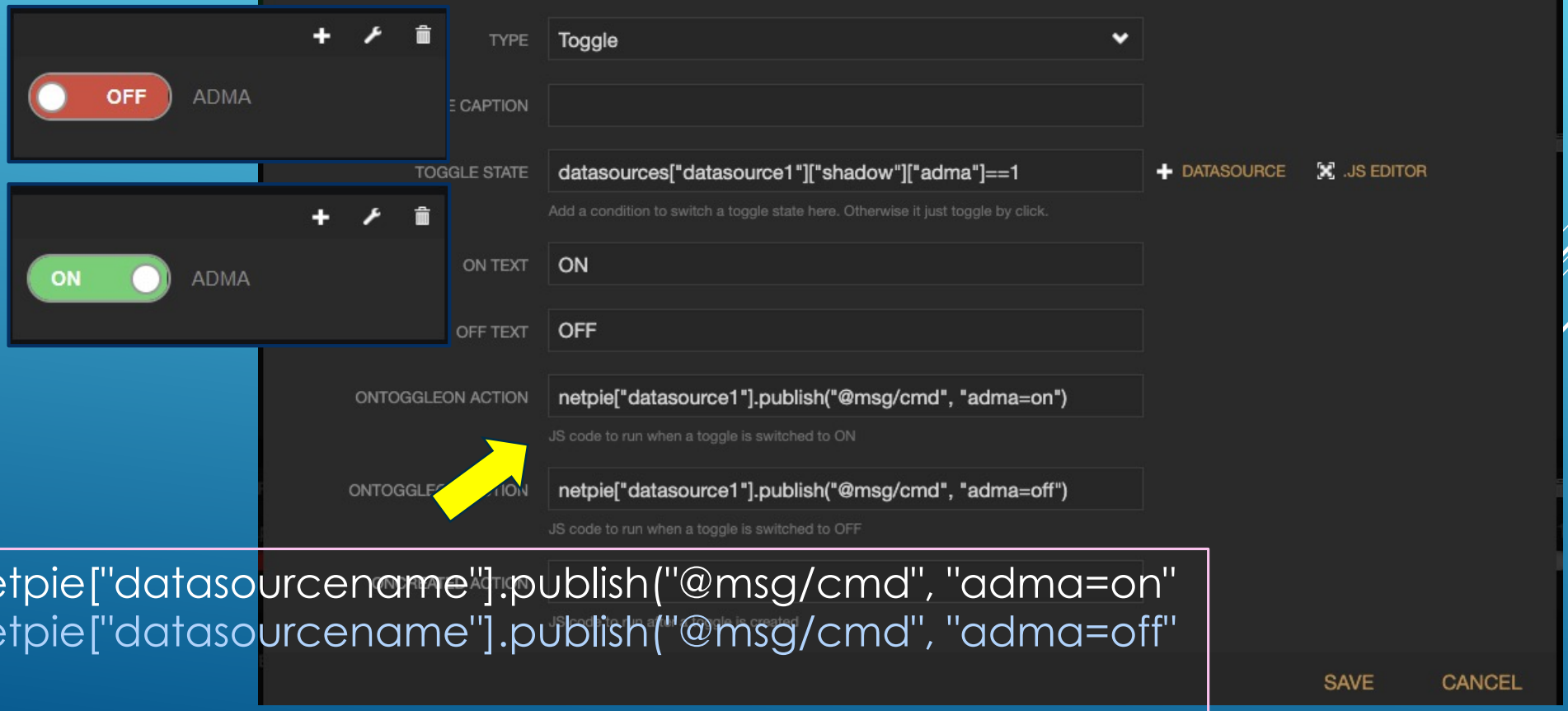
data written to NETPIE shadow



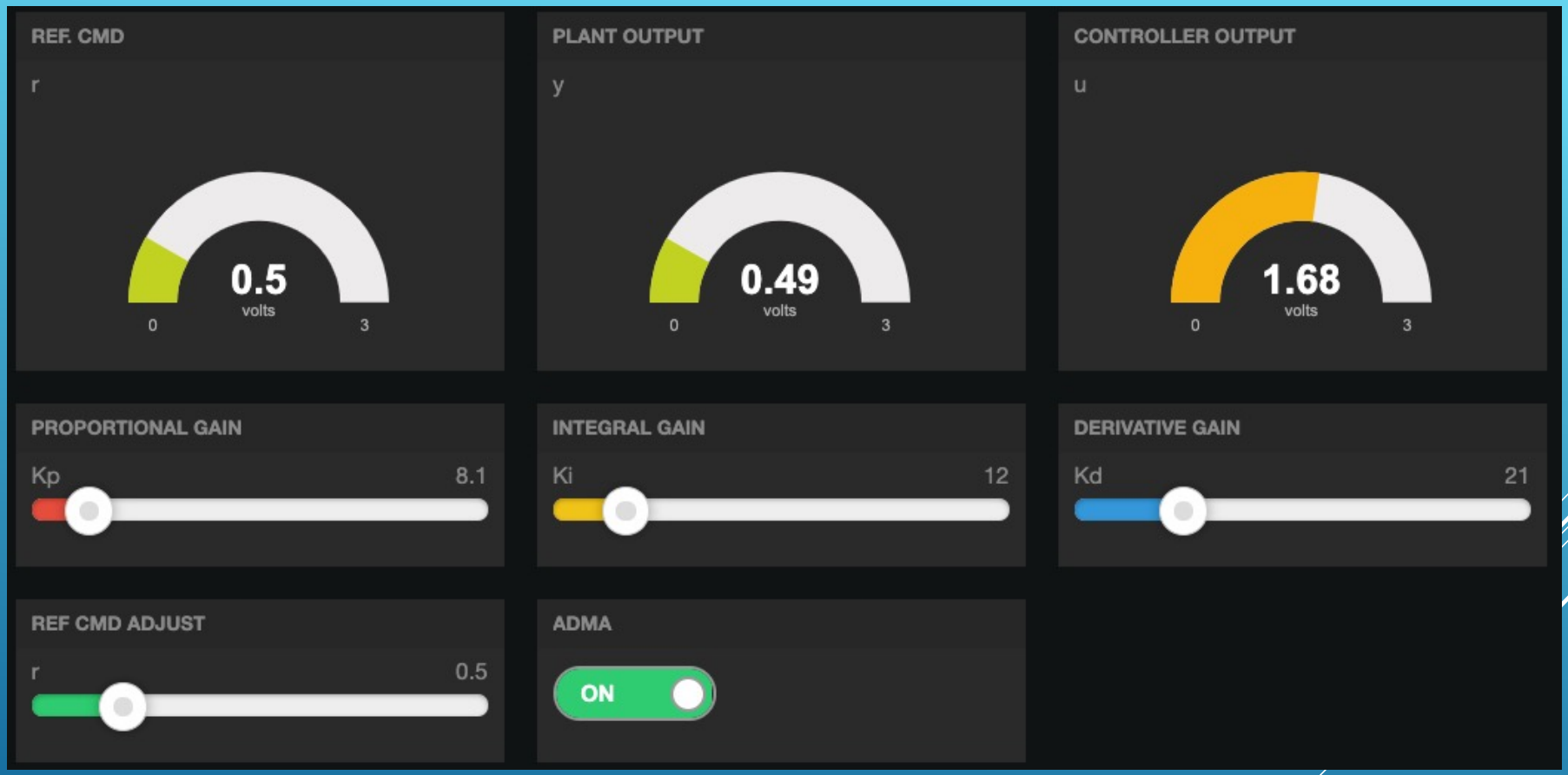
slider setup for proportional gain adjustment

netpie["datasourcename"].publish("@msg/cmd", "kp="+String(value))

ADMA toggle button setup




```
netpie["datasourcename"].publish("@msg/cmd", "adma=on"
netpie["datasourcename"].publish("@msg/cmd", "adma=off"
```



PID Freeboard