

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

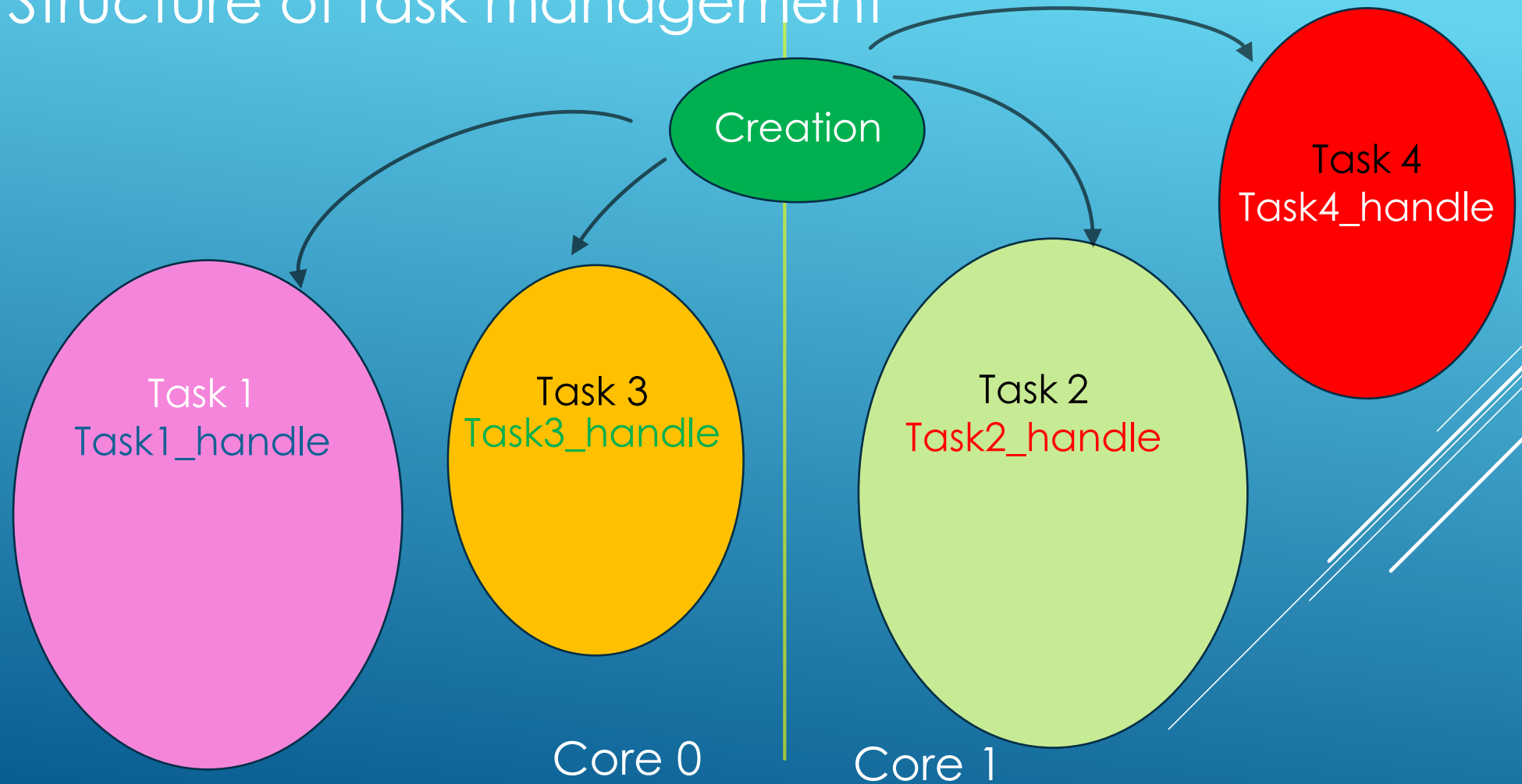
# FreeRTOS Basics

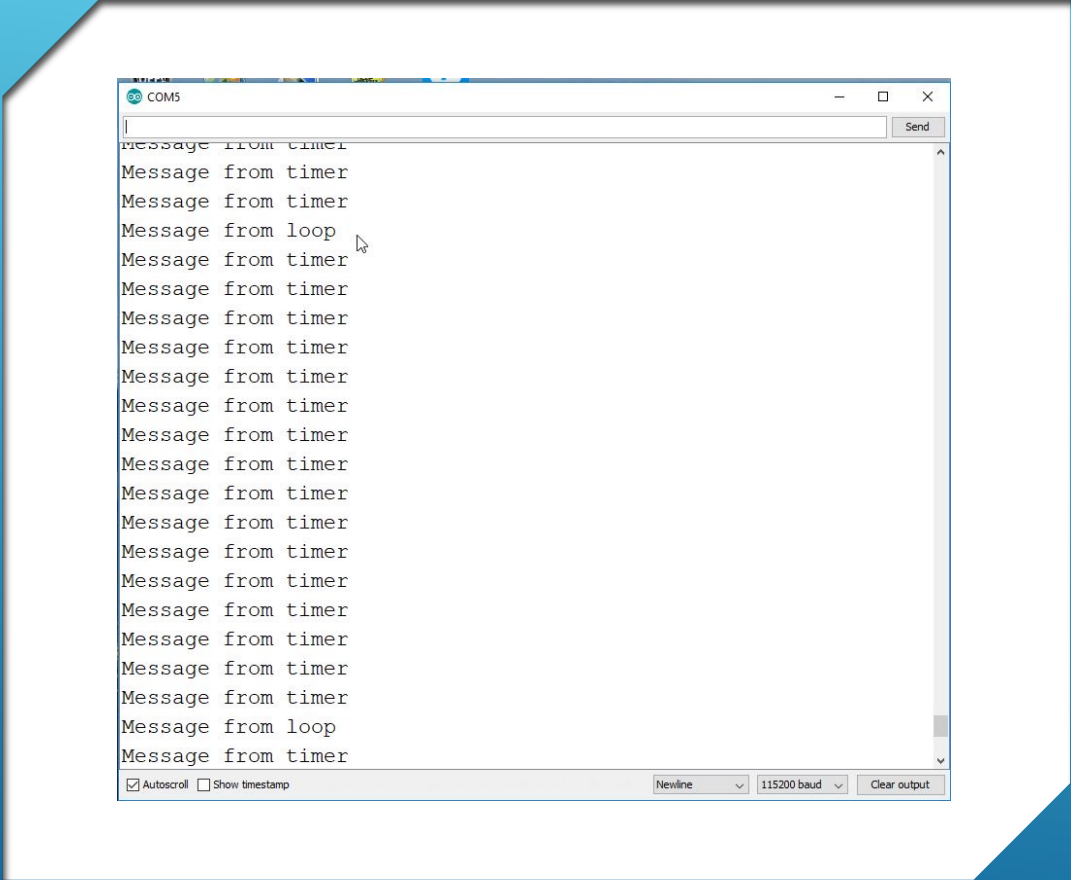
Dr.Varodom Toochinda

Dept. of Mechanical Engineering,  
Kasetsart University

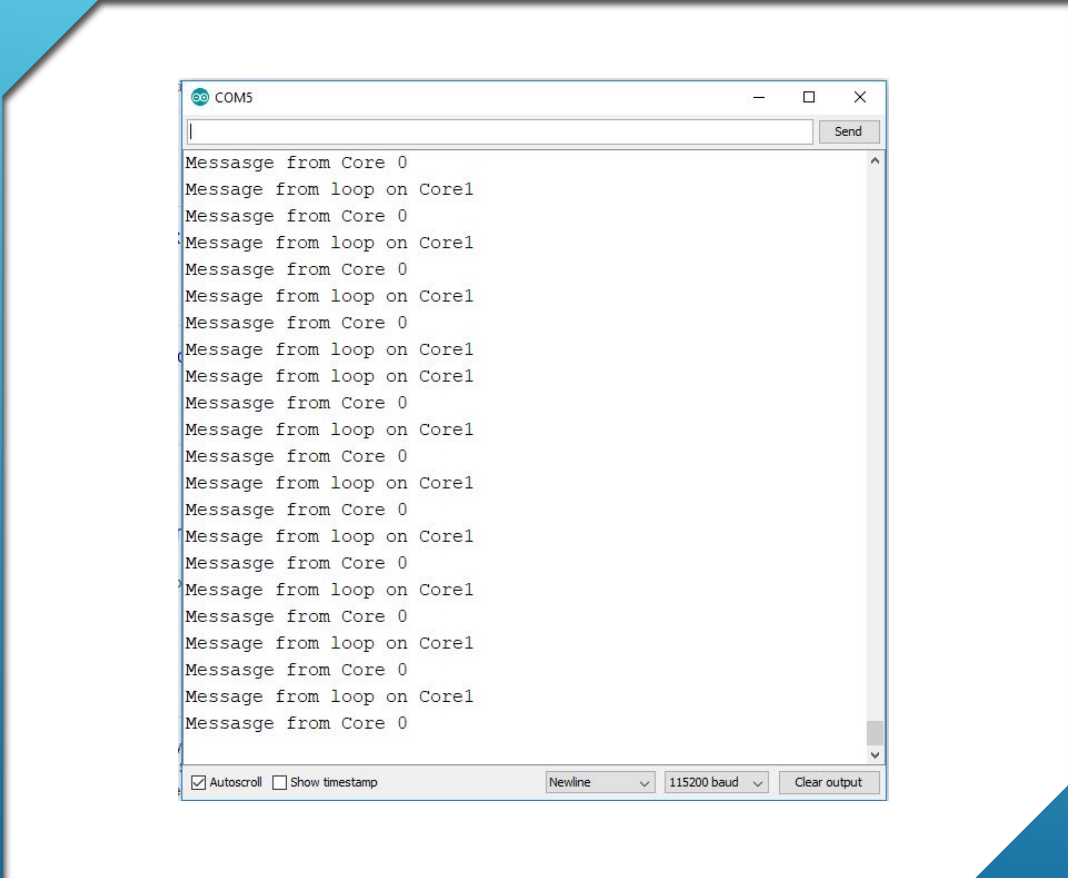
from chapter 8 of "IoT Development Handbook"

# Structure of task management





both executed on core 1



# 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 );  
}
```



```
COM4  
Message from task 1  
Message from task 2  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 2  
Message from task 1  
Ending task 2  
Ending task 1  
☒ Autoscroll
```

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<20000000; 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<20000000; ul++);  
    }  
    Serial.println("Ending task 2");  
    vTaskDelete( NULL );  
}
```





```
COM4  
|  
Message from task 1  
Message from task 2  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Message from task 1  
Ending task 1  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Message from task 2  
Ending task 2  
☒ Autoscroll
```

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 */
```



```
COM7

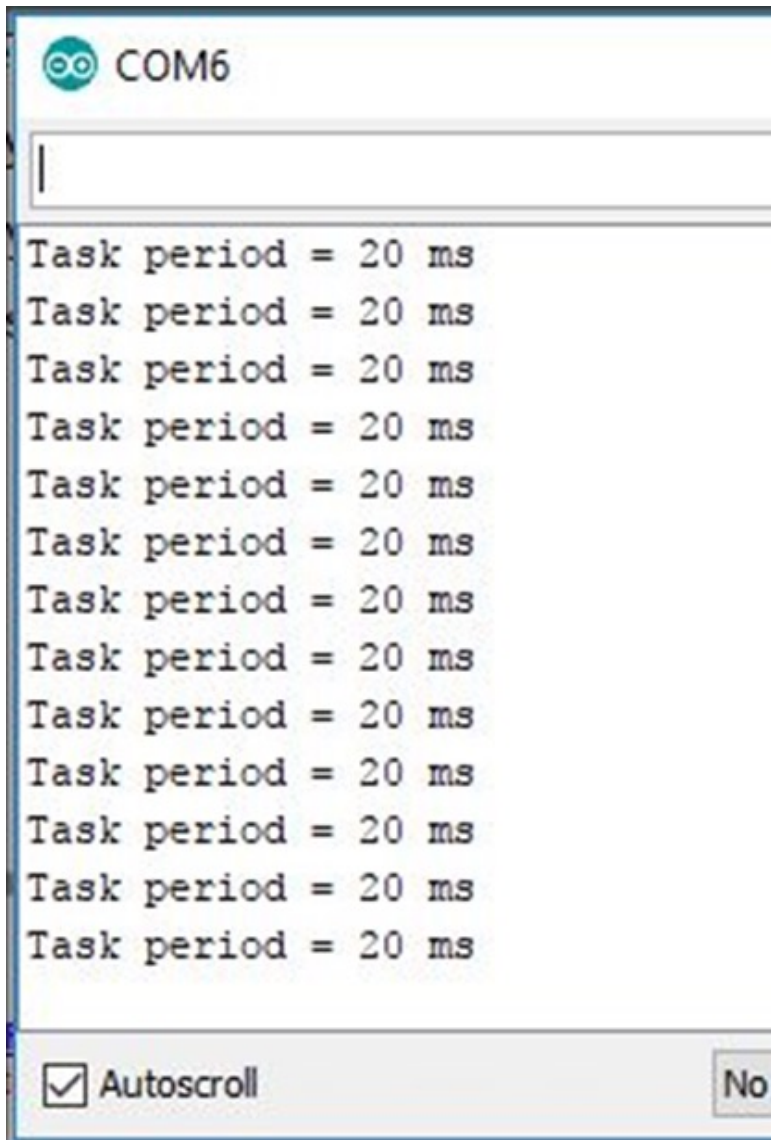
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Message from task 1 on core 1
Message from task 2 on core 0
Ending task 1
Ending task 2

☒ Autoscroll
```

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

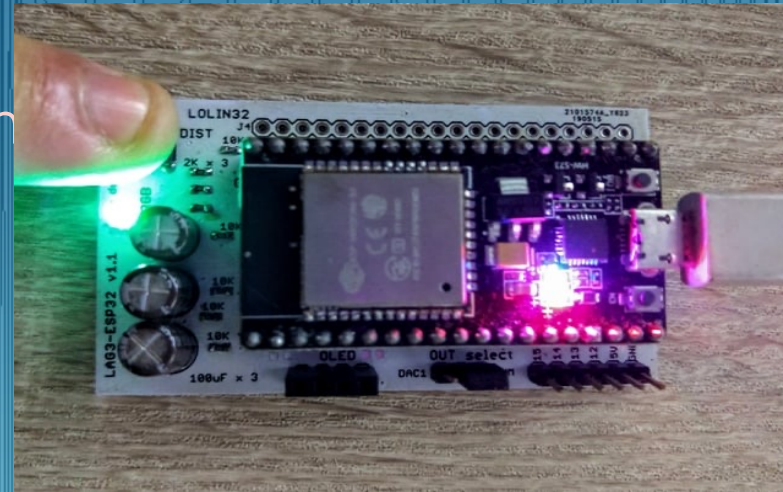
```
if (i==5){  
    Serial.println("Priority Changed: Task1=0, Task2=1");  
    vTaskPrioritySet(NULL, 0); // decrease task 1 priority  
    vTaskPrioritySet(xTask2Handle, 1); // increase task 2 priority  
}
```





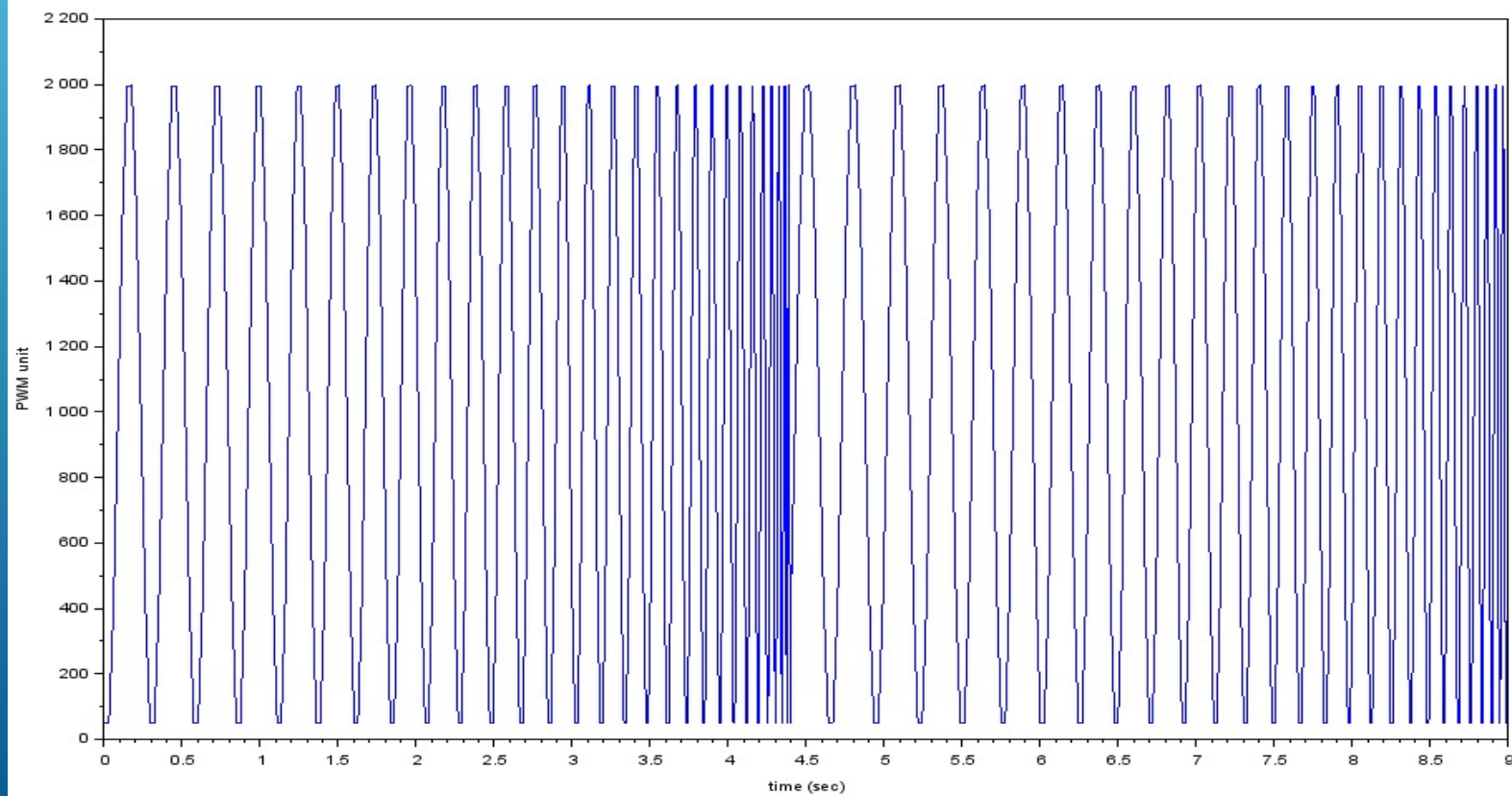
- ▶ Task 1 : blink LED
- ▶ Task 2 : generate chirp when switch is pressed
- ▶ Compare between
  - ▶ Multitask\_basic.ino
  - ▶ Multitask\_freeRTOS.ino

TRUE MULTI-TASKING





## 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");  
}
```

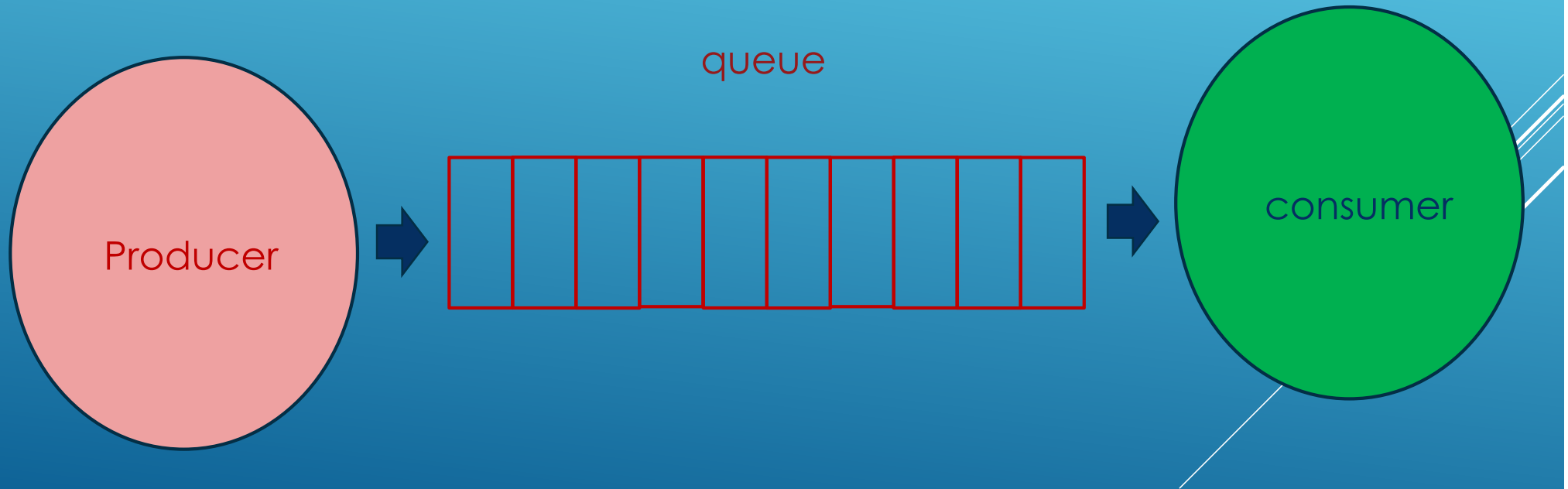
To write data to queue, use `xQueueSend()` with following format

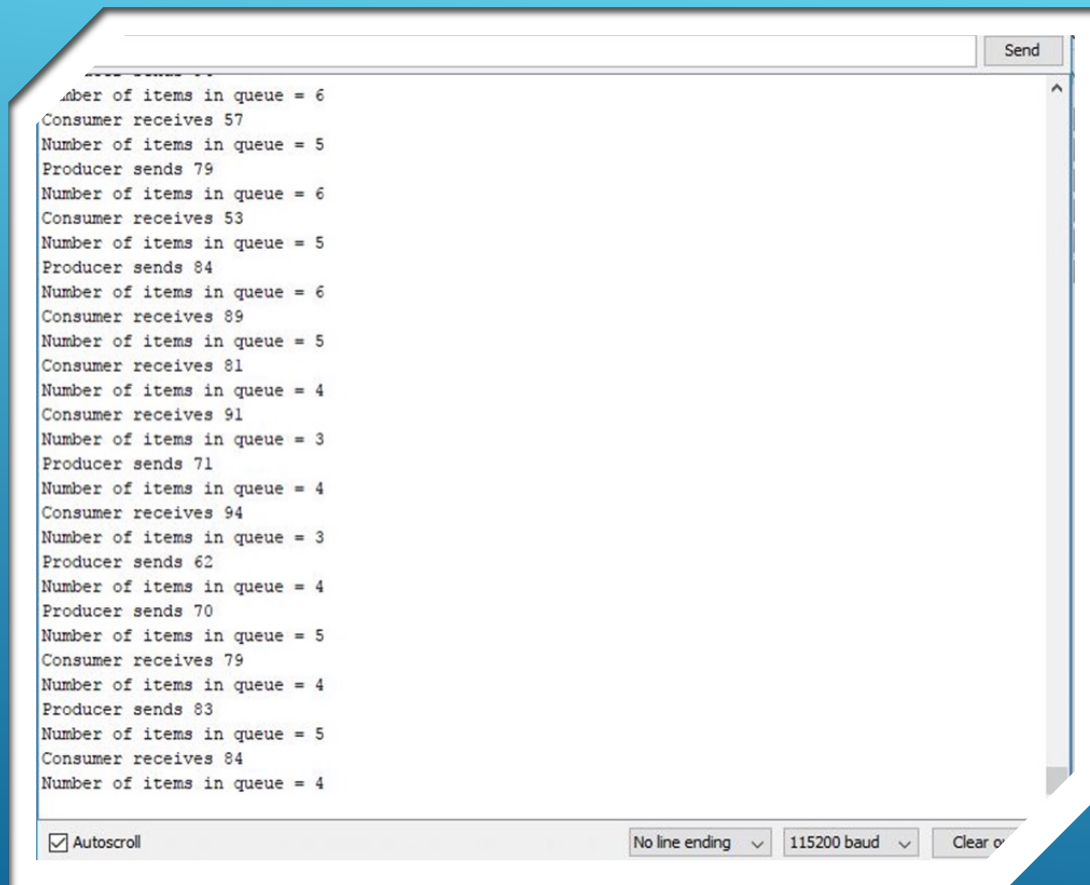
```
xQueueSend(queue, &data, xTicksToWait);
```

To read from queue, use

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

ex2\_6.ino : using queue





```
Number of items in queue = 6
Consumer receives 57
Number of items in queue = 5
Producer sends 79
Number of items in queue = 6
Consumer receives 53
Number of items in queue = 5
Producer sends 84
Number of items in queue = 6
Consumer receives 89
Number of items in queue = 5
Consumer receives 81
Number of items in queue = 4
Consumer receives 91
Number of items in queue = 3
Producer sends 71
Number of items in queue = 4
Consumer receives 94
Number of items in queue = 3
Producer sends 62
Number of items in queue = 4
Producer sends 70
Number of items in queue = 5
Consumer receives 79
Number of items in queue = 4
Producer sends 83
Number of items in queue = 5
Consumer receives 84
Number of items in queue = 4
```

☒ Autoscroll    No line ending    115200 baud    Clear

ex2\_6.ino  
output on  
serial  
monitor

“No different from periodic task”

ex2\_7.ino : timer task





use macro  
`taskENTER_CRITICAL()`  
and  
`taskEXIT_CRITICAL()`

stop scheduler or  
tasks that access  
protected variables

use semaphore or  
mutex

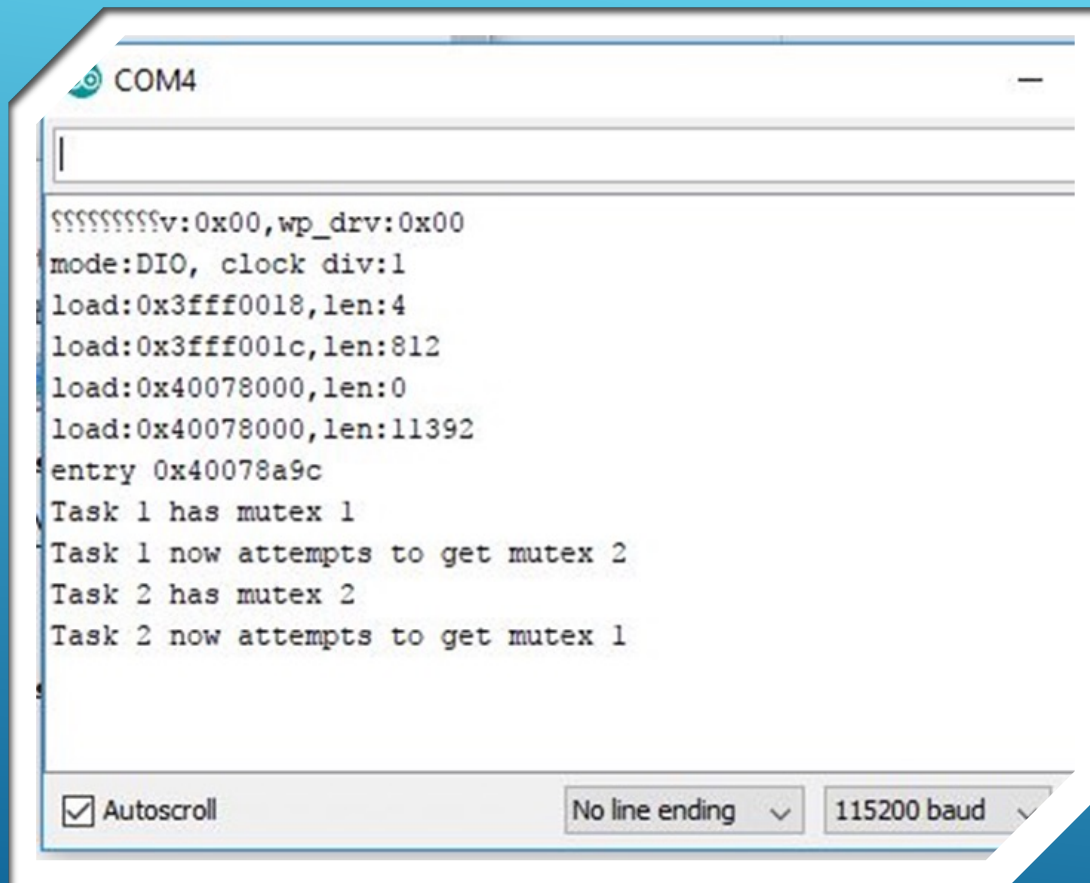
## Critical section protection

## 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 );
}
```

for



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 interpreter
5	1	NETPIE communication
7	1	Plant simulation

lag3\_freertos\_plantsim.ino

```
lag3_freertos_plantsim
388 // update previous values
389 e1 = e0; // true error
390 ed1 = ed0; // derivative term
391
392 ui1 = ui0; // integral term
393 ud1 = ud0; // derivative term
394
395 e0 = r_raw - y_raw; //
396 ep0 = wp*r_raw - y_raw; //
397 ed0 = wd*r_raw - y_raw; //
398
399 if (controltype==PID) {
400     up0 = kp*ep0; // output of
401     ui0 = ui1 +bi*(e0+e1) + kt*
402     ud0 = ad*ud1 + bd*(ed0-ed1)
403     u0 = up0 + ui0 + ud0;
404 }
405 else { // controltype ==

Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0
Hash of data verified.
```

```
stepdata=0.3]
datamat = np.array([
[0.08, 0.30, 0.10, 3.30],
[0.16, 0.30, 0.10, 1.73],
[0.24, 0.30, 0.10, 1.33],
[0.32, 0.30, 0.10, 1.30],
[0.40, 0.30, 0.11, 1.16],
[0.48, 0.30, 0.12, 1.20],
[0.56, 0.30, 0.12, 1.15],
[0.64, 0.30, 0.13, 1.12],
[0.72, 0.30, 0.14, 1.04],
[0.80, 0.30, 0.16, 1.01],
[0.88, 0.30, 0.17, 0.96],
[0.96, 0.30, 0.18, 0.82],
[1.04, 0.30, 0.20, 0.80],
[1.12, 0.30, 0.21, 0.71],
[1.20, 0.30, 0.23, 0.64],
[1.28, 0.30, 0.25, 0.58],
[1.36, 0.30, 0.26, 0.51],
[1.44, 0.30, 0.28, 0.49],
[1.52, 0.30, 0.29, 0.33],
[1.60, 0.30, 0.31, 0.33],
[1.68, 0.30, 0.32, 0.31],
[1.76, 0.30, 0.34, 0.22],
[1.84, 0.30, 0.35, 0.19],
```

☒ Autoscroll ☐ Show timestamp

step data  
on serial  
monitor

Jupyter plot\_pid\_resp Last Checkpoint: Last Friday at 3:42 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [3]: def pid_compare(kp, ki, kd, datamat, scale=1):

    # extract vectors from datamat
    tvec = datamat[:,0]
    rvec = datamat[:,1]
    yvec = datamat[:,2]
    uvec = datamat[:,3]

    s = ctl.tf('s')

    # 3rd-order lag plant
    P = 1/((s+1)**3)

    # continuous-time PID
    N = 20
    C_c = kp + ki/s + s*N*kd/(s+N)
    L_c = C_c*P
    sys_c = ctl.feedback(L_c)

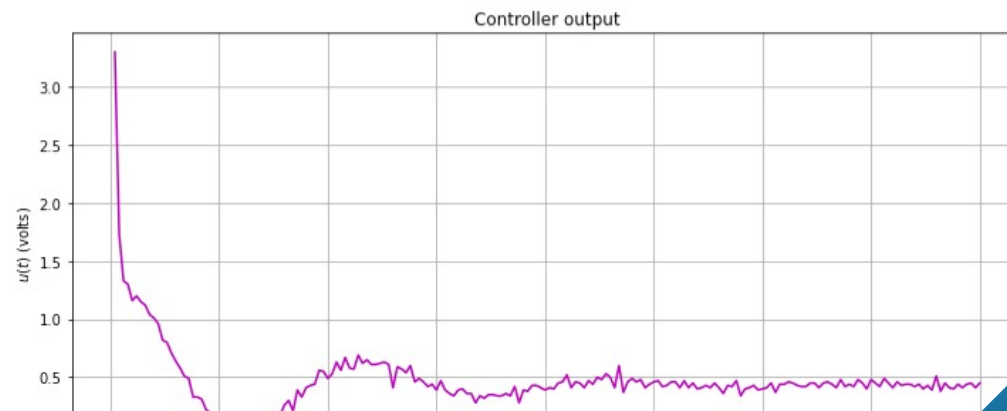
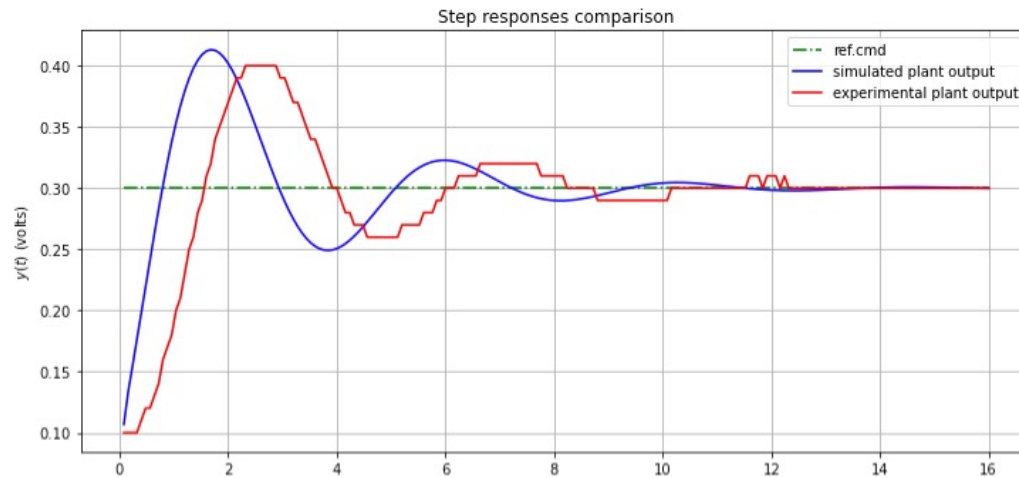
    # simulate response
    Tv, y_c = ctl.step_response(sys_c, tvec, X0=0.1/scale)

    # plot the comparison
    fig, (ax1, ax2) = plt.subplots(2, figsize=(12,12))
    #fig.suptitle('Step responses comparison')
    ax1.set_title('Step responses comparison')
    ax1.plot(tvec, rvec, 'g-.', tvec, scale*y_c, 'b-', tvec, yvec, 'r-')
    #ax1.set_xlabel('time (sec)')
    ax1.set_ylabel('$y(t)$ (volts)')
    ax1.legend(['ref.cmd', 'simulated plant output', 'experimental plant output'])
    ax1.grid(True)

    ax2.plot(tvec, uvec, 'm-')
    ax2.set_xlabel('time (sec)')
    ax2.set_ylabel('$u(t)$ (volts)')
    ax2.grid(True)
    ax2.set_title('Controller output')
    plt.show()
```

write Python  
function to  
compare  
step  
responses

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



step  
response  
compare  
result

The screenshot displays the NETPIE web interface. On the left is a sidebar with navigation links: Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main content area shows the breadcrumb 'project1 / device / device1'. It features a 'Description' section, a 'Key' section with fields for Client ID, Token, Secret, and Status (Online), and tabs for Shadow, Schema, and Trigger. The 'Shadow' tab is active, showing a tree view of a node 'object {11}' with various parameters like adma, controller, kd, ki, kp, kt, r, u, wd, wp, and y, each with a numerical value.

NETPIE

project1

project1 / device / device1

**Description**

**Key**

Client ID : 603dded2-bff  
Token : 9kLpZBB3yc3  
Secret : fRY()Sy07lm)z  
Status : ● Online

Shadow Schema Trigger

Tree


Select a node...

- object {11}
  - adma : 1
  - controller : 0
  - kd : 21
  - ki : 12
  - kp : 8.1
  - kt : 0
  - r : 0.5
  - u : 1.52
  - wd : 1
  - wp : 1
  - y : 0.49

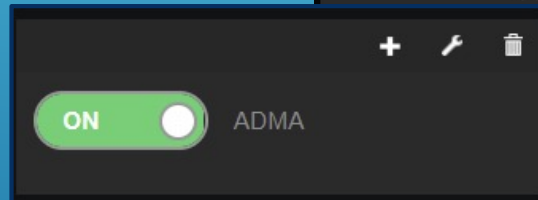
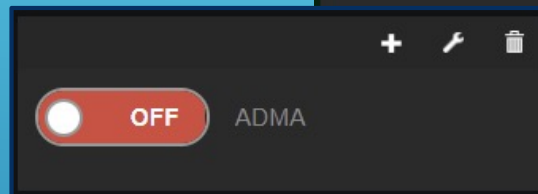
data  
written to  
NETPIE  
shadow



# slider setup for proportional gain adjustment

SLIDER CAPTION	kp
FILLED COLOR	Red
DISPLAY VALUE	YES 
MIN VALUE	0
MAX VALUE	100
STEP	0.1
INITIAL VALUE	0
The default value set only the first time the widget is loaded.	
AUTO UPDATED VALUE	<code>datasources["datasource1"]["shadow"]["kp"]</code> + DATASOURCE JS EDITOR
Slider will be updated upon the change of variables (e.g. other data sources).	
ONSTART ACTION	+ DATASOURCE JS EDITOR
Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.	
ONSLIDE ACTION	+ DATASOURCE JS EDITOR
Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.	
ONSTOP ACTION	<code>netpie["datasource1"].publish("@msg/cmd", "kp="+String(value))</code> + DATASOURCE JS EDITOR
Add some Javascript here. You can access to a slider attribute using variables 'value' and 'percent'.	
ONCREATED ACTION	<code>netpie["datasourcename"].publish("@msg/cmd", "kp="+String(value))</code>

# ADMA toggle button setup



A simple toggle widget that can perform javascript action.

TYPE **Toggle**

DESCRIPTION

TOGGLE STATE `datasources["datasource1"]["shadow"]["adma"]==1` [+ DATASOURCE](#) [.JS EDITOR](#)

Add a condition to switch a toggle state here. Otherwise it just toggle by click.

ON TEXT **ON**

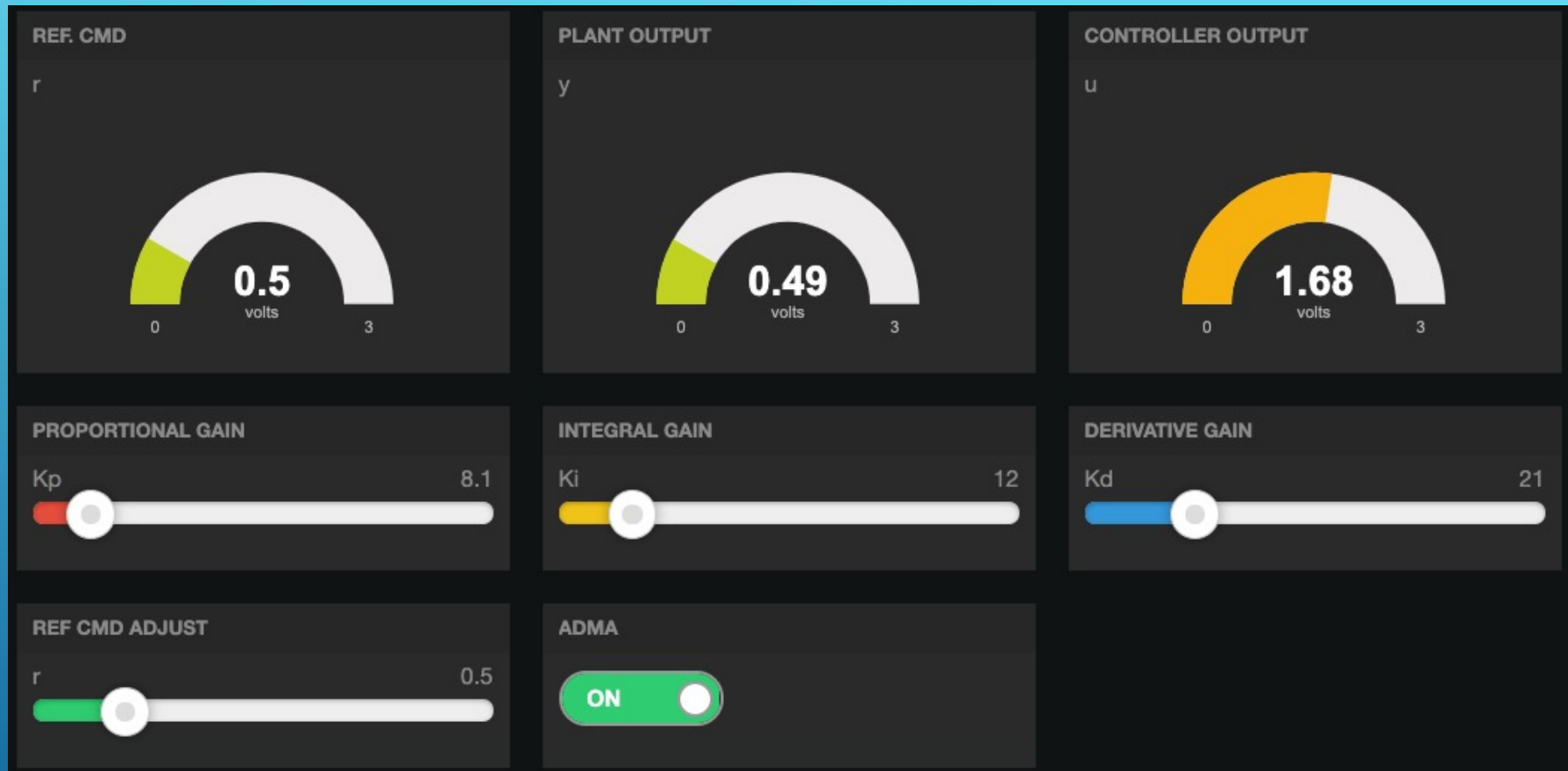
OFF TEXT **OFF**

ONTOGGLEON ACTION `netpie["datasource1"].publish("@msg/cmd", "adma=on")`  
JS code to run when a toggle is switched to ON

ONTOGGLEOFF ACTION `netpie["datasource1"].publish("@msg/cmd", "adma=off")`  
JS code to run when a toggle is switched to OFF

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

SAVE CANCEL



## PID Freeboard