Program SPC diupload ke **Arduino Mega 2560** dengan **Ethernet Shield**.

# Library

**##Library yang digunakan**Terdapat 21 Library

## **Tiga** buah Library untuk **Ethernet Shield**

|  |
| --- |
| #include <SPI.h>  #include <Ethernet.h>  #include <PubSubClient.h> |

## **Dua** buah Library untuk **LCD 20x4**

|  |
| --- |
| #include <Wire.h> // Comes with Arduino IDE  #include <LiquidCrystal\_I2C.h> |

Library <LiquidCrystal\_I2C.h> diperoleh dari <https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads/> yaitu dengan mendownload dan mengimport file **Newliquidcrystal\_1.3.5.zip** ke Arduino IDE. Library <LiquidCrystal\_I2C.h> yang sebelumnya diperoleh dan diinstall dari sumber lain harus dihapus terlebih dahulu dari folder C:\Users\[nama user]\Documents\Arduino\libraries

## **Satu** buah Library untuk Kalkulasi RPM

|  |
| --- |
| #include "Syergie.h" |

Terdapat perubahan minor yang dilakukan di file Syergie.h yaitu dengan mengubah line 14 yang semula:  
 int Proximity**::**calcRPM**();**  
menjadi:  
 int calcRPM**();**  
dan perubahan nilai di Syergie.cpp pada line 30 yang semula:  
 rpm **=** **(**312000000**/(**duration**));** // rpm = (1/ time millis)\*1000\*1000\*60;  
menjadi:  
 rpm **=** **(**60000000**/(**duration**));** // rpm = (1/ time millis)\*1000\*1000\*60;

## **Lima belas** buah Library untuk **FreeRTOS**

Sebenarnya tidak diperlukan semua, namun supaya lengkap dimasukkan semua.

|  |
| --- |
| #include <Arduino\_FreeRTOS.h>  #include <croutine.h>  #include <event\_groups.h>  #include <FreeRTOSConfig.h>  #include <FreeRTOSVariant.h>  #include <list.h>  #include <mpu\_wrappers.h>  #include <portable.h>  #include <portmacro.h>  #include <projdefs.h>  #include <queue.h>  #include <semphr.h>  #include <stack\_macros.h>  #include <task.h>  #include <timers.h> |

Library Arduino\_FreeRTOS dapat langsung diinstall melalui Library Manager Arduino IDE dengan mensearch FreeRTOS.

# Definisi-Definisi

## Definisi Pin-Pin

Definisi Pin-Pin dibuat di awal supaya program lebih readable.

|  |
| --- |
| #define pwm\_right 3  #define pwm\_left 5  #define pinRPM\_1 A1  #define pinRPM\_2 A2 |

## Definisi LCD

|  |
| --- |
| LiquidCrystal\_I2C lcd**(**0x27**,** 2**,** 1**,** 0**,** 4**,** 5**,** 6**,** 7**,** 3**,** POSITIVE**);** // Set the LCD I2C address |

## Mendefinisikan prototype Task-Task

|  |
| --- |
| void TaskRpmMeasurement**(** void **\***pvParameters **);**  void TaskMQTT**(** void **\***pvParameters **);**  void TaskPID**(** void **\***pvParameters **);**  void TaskLCD**(** void **\***pvParameters **);**  void TaskRPM**(** void **\***pvParameters **);** |

## Mendefinisikan variabel network Ethernet Shield

|  |
| --- |
| byte mac**[]** **=** **{** 0xDE**,** 0xED**,** 0xBA**,** 0xFE**,** 0xFE**,** 0xED **};**  IPAddress ip**(**192**,** 168**,** 1**,** 100**);**  IPAddress server**(**192**,** 168**,** 1**,** 2**);**  EthernetClient ethClient**;**  PubSubClient client**(**ethClient**);** |

## Mendefinisikan variabel-variabel global yang digunakan oleh program

|  |
| --- |
| /\* Fuel Tank Level, Steer, and Depth\*/  int level\_tanki **=** 0**;**  int steer **=** 0**;**  int depth **=** 0**;**  //Buffers for Publishing via MQTT  char send\_rpm\_engine**[**10**];**  char send\_rpm\_propeller**[**10**];**  char send\_tanki**[**10**];**  char send\_Speed**[**10**];**  //Bufers for Displaying to LCD  char disp\_rpm\_engine**[**5**];**  char disp\_rpm\_propeller**[**5**];**  char disp\_tanki**[**3**];**  char disp\_steer**[**4**];**  char disp\_depth**[**4**];**  /\* RPM Measurement \*/  Proximity speed\_1**;**  Proximity speed\_2**;**  int rpm\_1 **=** 0**;**  int rpm\_2 **=** 0**;**  int Speed **=** 1500**;**  //PID CONSTANTS  double P **=** 3.0209**;**  double I **=** 0.0000293**;**  double D **=** 0.2353**;**  int LPWM **=** 12**;**  int RPWM **=** 5**;**  int RPWM1 **=** 3**;**  int LPWM1 **=** 9**;**  int RPWM2 **=** 2**;**  int LPWM2 **=** 4**;**  int RPWM3 **=** 10**;**  int LPWM3 **=** 12**;**  int L\_EN **=** 8**;**  int R\_EN **=** 7**;**  int pwm**;** //???  int pwm1**;**  int pwm2**;**  int pwm3**;**  double Buff**;** // posisi baru  double Last **=** 0**;** // posisi lama  double PTerm**;**  double ITerm**;**  double DTerm**;**  double PIDTerm**;**  double Ts **=** 1**;**  double cepat **=** 0**;**  double position\_out **=** 0 **;**  double position\_in **=** 500**;**  double messi**;**  double Error**;** // perbedaan feedback dengan input  double Sum **=** 0**;** // hasil integral error  //END OF PID CONSTANTS |

## Mendefinisikan prosedur callback

|  |
| --- |
| void callback**(**char**\*** topic**,** byte**\*** payload**,** unsigned int length**)** **{**  Serial**.**print**(**F**(**"Message arrived ["**));**Serial**.**print**(**topic**);**Serial**.**print**(**F**(**"] "**));**    int value **=** 0**;**  **for** **(**int i**=**0**;**i**<**length**;**i**++)** **{** //storing payload to integer value  Serial**.**print**((**char**)**payload**[**i**]);**  value **=** value**\***10**;**  value**=**value**+(**payload**[**i**]-**48**);**  **}**  Serial**.**println**();**    **if(**topic**[**1**]==**'p'**){**  client**.**publish**(**"confirm\_SPC1"**,**"Received"**);**  Speed **=** value**;**  // Serial.print(F("value : "));Serial.println(value);  // Serial.print(F("Speed : "));Serial.println(Speed);  // Serial.print(F("PWM 1 : "));Serial.println(pwm1);  // Serial.print(F("PWM 2 : "));Serial.println(pwm2);  **}**    **else** **if(**topic**[**0**]==**'M'**){**  client**.**publish**(**"1EngineSpeed"**,**dtostrf**(**rpm\_1**,** 4**,** 0**,** send\_rpm\_engine**));**  client**.**publish**(**"1PropellerSpeed"**,**dtostrf**(**rpm\_2**,** 4**,** 0**,** send\_rpm\_propeller**));**  client**.**publish**(**"1Fuel"**,**dtostrf**(**level\_tanki**,** 2**,** 0**,** send\_tanki**));**  client**.**publish**(**"1Box"**,**"connected"**);**  **}**  **}** |

## Mendefinisikan prosedur reconnect

|  |
| --- |
| void reconnect**()** **{**  // Loop until we're reconnected  **while** **(!**client**.**connected**())** **{**  Serial**.**print**(**F**(**"Attempting MQTT connection... "**));**  // Attempt to connect  **if** **(**client**.**connect**(**"Box1"**))** **{**  Serial**.**print**(**server**);**  Serial**.**println**(**F**(**" connected"**));**  // Once connected, publish an announcement...    // ... and resubscribe  client**.**subscribe**(**"1pulse\_spc"**);**  client**.**subscribe**(**"MainControl"**);**  **}** **else** **{**  Serial**.**print**(**F**(**"failed, rc="**));**  Serial**.**print**(**client**.**state**());**  Serial**.**println**(**F**(**" try again in 5 seconds"**));**  // Wait 5 seconds before retrying  delay**(**5000**);**  **}**  **}**  **}** |

# Masuk ke Program Arduino

## Setup

|  |
| --- |
| // the setup function runs once when you press reset or power the board  void setup**()** **{**  speed\_1**.**pinRPM**(**pinRPM\_1**);**  speed\_2**.**pinRPM**(**pinRPM\_2**);**    // initialize serial communication at 115200 bits per second:  Serial**.**begin**(**115200**);**    **while** **(!**Serial**)** **{**  **;** // wait for serial port to connect. Needed for native USB, on LEONARDO, MICRO, YUN, and other 32u4 based boards.  **}**  client**.**setServer**(**server**,** 1883**);**  client**.**setCallback**(**callback**);**  Ethernet**.**begin**(**mac**,** ip**);**    // Allow the hardware to sort itself out  delay**(**1500**);**    lcd**.**begin**(**20**,**4**);** // initialize the lcd for 20 chars 4 lines, turn on backlight  // ------- Quick 3 blinks of backlight -------------  **for(**int i **=** 0**;** i**<** 3**;** i**++)**  **{**  lcd**.**backlight**();**  delay**(**250**);**  lcd**.**noBacklight**();**  delay**(**250**);**  **}**  lcd**.**backlight**();** // finish with backlight on  //-------- Write characters on the display ------------------  // NOTE: Cursor Position: Lines and Characters start at 0  lcd**.**setCursor**(**4**,**0**);** //Start at character 4 on line 0  lcd**.**print**(**"Salam KAPAL!"**);**  delay**(**1000**);**  // Now set up two tasks to run independently.  xTaskCreate**(**  TaskRpmMeasurement  **,** **(**const portCHAR **\*)** "RpmMeasurement" // A name just for humans  **,** 1024 // This stack size can be checked & adjusted by reading the Stack Highwater, 128  **,** **NULL**  **,** 1 // Priority, with 3 (configMAX\_PRIORITIES - 1) being the highest, and 0 being the lowest.  **,** **NULL** **);**  xTaskCreate**(**  TaskMQTT  **,** **(**const portCHAR **\*)** "MQTT"  **,** 1024 // Stack size  **,** **NULL**  **,** 1 // Priority  **,** **NULL** **);**  xTaskCreate**(**  TaskPID  **,** **(**const portCHAR **\*)** "PID"  **,** 1024 // Stack size  **,** **NULL**  **,** 1 // Priority  **,** **NULL** **);**    xTaskCreate**(**  TaskLCD  **,** **(**const portCHAR **\*)** "LCD"  **,** 1024 // Stack size  **,** **NULL**  **,** 1 // Priority  **,** **NULL** **);**    xTaskCreate**(**  TaskRPM  **,** **(**const portCHAR **\*)** "RPM"  **,** 1024 // Stack size  **,** **NULL**  **,** 1 // Priority  **,** **NULL** **);**  // Now the task scheduler, which takes over control of scheduling individual tasks, is automatically started.  **}** |

## Loop

|  |
| --- |
| void loop**(){}** // Empty. Things are done in Tasks. |

## Task-Task

### Task for RPM and Measurements

|  |
| --- |
| /\*---------------------- Tasks ---------------------\*/  void TaskRpmMeasurement**(**void **\***pvParameters**)** // Task for RPM and Measurements  **{**  **(**void**)** pvParameters**;**  TickType\_t xLastWakeTime**;**  xLastWakeTime **=** xTaskGetTickCount**();**    int countA**,** countB**;**  **for** **(;;)** // A Task shall never return or exit.  **{**  countB **=** millis**()** **-** countA**;**  // Serial.print(F("Time TaskRPM: "));Serial.println(countB);    rpm\_1 **=** speed\_1**.**calcRPM**();**  rpm\_2 **=** speed\_2**.**calcRPM**();**  countA **=** millis**();**  // rpm\_1 += 100;  // rpm\_2 += 100;  // if (rpm\_1 > 9999) {  // rpm\_1 = 0;  // }  // if (rpm\_2 > 9999){  // rpm\_2 = 0;  // }  steer **+=** 10**;**  **if** **(**steer **>** 360**)** **{**  steer **=** 0**;**  **}**  depth **+=** 13**;**  **if** **(**depth **>** 99**)** **{**  depth **=** 0**;**  **}**    Serial**.**print**(**F**(**"RPM 1 : "**));**Serial**.**println**(**rpm\_1**);**  Serial**.**print**(**F**(**"RPM 2 : "**));**Serial**.**println**(**rpm\_2**);**    //Fuel Measurement  // level\_tanki = map(analogRead(A1),0,1023,0,20);  level\_tanki **+=** 17**;**  **if** **(**level\_tanki **>** 99**)** **{**  level\_tanki **=** 0**;**  **}**  // Serial.print(F("Level Tangki : "));Serial.println(level\_tanki);    //Governor Control  **switch(**Speed**){**  **case** 1900**:** pwm1 **=** 200**;** pwm2 **=** 0**;** **break;** //Speed UP  **case** 1500**:** pwm1 **=** 0**;** pwm2 **=** 0**;** **break;** //Steady  **case** 1300**:** pwm1 **=** 0**;** pwm2 **=** 200**;** **break;** //Speed Down  **}**  Serial**.**print**(**F**(**"Speed : "**));**Serial**.**println**(**Speed**);**  Serial**.**print**(**F**(**"PWM 1 : "**));**Serial**.**println**(**pwm1**);**  Serial**.**print**(**F**(**"PWM 2 : "**));**Serial**.**println**(**pwm2**);**  analogWrite**(**pwm\_right**,** pwm1**);**  analogWrite**(**pwm\_left**,** pwm2**);**    vTaskDelayUntil**(** **&**xLastWakeTime**,** 1**);**//pdMS\_TO\_TICKS( 120 ) );//( 300 / portTICK\_PERIOD\_MS ) ); // wait for one second  **}**  **}** |

### Task MQTT

|  |
| --- |
| void TaskMQTT**(**void **\***pvParameters**)** // Task MQTT  **{**  **(**void**)** pvParameters**;**  TickType\_t xLastWakeTime**;**  xLastWakeTime **=** xTaskGetTickCount**();**    int countC**,** countD**;**  **for** **(;;)**  **{**  countD **=** millis**()** **-** countC**;** countC **=** millis**();**  // Serial.print(F("Time TaskMQT: "));Serial.println(countD);    rpm\_1 **=** speed\_1**.**calcRPM**();**  rpm\_2 **=** speed\_2**.**calcRPM**();**  **if** **(!**client**.**connected**())** **{**  reconnect**();**  **}**  client**.**loop**();**    vTaskDelayUntil**(** **&**xLastWakeTime**,** 10**);**//pdMS\_TO\_TICKS( 120 ) );//( 300 / portTICK\_PERIOD\_MS ) ); // wait for one second  **}**  **}** |

### Task PID

|  |
| --- |
| void TaskPID**(**void **\***pvParameters**)** // Task PID  **{**  **(**void**)** pvParameters**;**  TickType\_t xLastWakeTime**;**  xLastWakeTime **=** xTaskGetTickCount**();**    int countE**,** countF**;**  **for** **(;;)**  **{**  countF **=** millis**()** **-** countE**;** countE **=** millis**();**  // Serial.print(F("Time TaskPID: "));Serial.println(countF);  rpm\_1 **=** speed\_1**.**calcRPM**();**  rpm\_2 **=** speed\_2**.**calcRPM**();**  double position\_out\_1 **=** 0**;**  double position\_out **=** analogRead**(**A1**);**    // Perhitungan PID  Error **=** position\_in **-** position\_out**;**  Buff **=** position\_out**;** // untuk mencari derivatif  Sum **=** Sum **+** Error**;** // hasil integral dari error  PTerm **=** Error**\***P**;** // Proporsional  //ITerm = Sum\*I\*Ts; // Integral  DTerm **=** D**\*(**Last **-** Buff**)/**Ts**;** // Derivatif  PIDTerm **=** PTerm **+** DTerm**;** // total PID    **if** **(**PIDTerm **>=** 255**)**  PIDTerm **=** 255**;**  **if** **(**PIDTerm **<=** **-**255**)**  PIDTerm **=** **-**255**;**    cepat **=** PIDTerm**;** // Hasil PID dijadikan data kecepatan    **if** **(**pwm **>** 255**){**  pwm **=** 255**;**  **}**  **else** **if** **(**pwm **<-**255**){**  pwm **=** **-**255**;**  **}**    **if** **(**pwm2 **>** 255**){**  pwm2 **=** 255**;**  **}**  **else** **if** **(**pwm **<-**255**){**  pwm2 **=** **-**255**;**  **}**    **if** **(**pwm3 **>** 255**){**  pwm3 **=** 255**;**  **}**  **else** **if** **(**pwm3 **<-**255**){**  pwm3 **=** **-**255**;**  **}**    // Jika nilai kecepatan (-), stir berputar ke kanan  **if** **(**cepat **<** 0**){**  // putar kanan  int reversePWM **=** **-**cepat**;**  analogWrite**(**11**,** 0**);**  analogWrite**(**12**,** reversePWM**);**  **}**  // Jika nilai kecepatan (+), stir berputar ke kanan  **else** **if** **(**cepat **>=** 0**)**  **{**  // putar kiri  int forwardPWM **=** cepat**;**  analogWrite**(**11**,** forwardPWM**);**  analogWrite**(**12**,** 0**);**  **}**  Last **=** position\_out**;** // untuk mencari derivatif  //vTaskDelay(Ts\*1);  // Delay sebesar 10 tick  vTaskDelayUntil**(** **&**xLastWakeTime**,** Ts**\***10**);**//pdMS\_TO\_TICKS( 120 ) );//Ts\*300 / portTICK\_PERIOD\_MS ) ); // wait for one second  **}**  **}** |

### Task LCD

|  |
| --- |
| void TaskLCD**(**void **\***pvParameters**)** // This is a task.  **{**  **(**void**)** pvParameters**;**  TickType\_t xLastWakeTime**;**  xLastWakeTime **=** xTaskGetTickCount**();**    int countG**,** countH**;**  **for** **(;;)**  **{**  countH **=** millis**()** **-** countG**;** countG **=** millis**();**  // Serial.print(F("Time TaskLCD: "));Serial.println(countH);  rpm\_1 **=** speed\_1**.**calcRPM**();**  rpm\_2 **=** speed\_2**.**calcRPM**();**  dtostrf**(**rpm\_1**,** 4**,** 0**,** disp\_rpm\_engine**);**  dtostrf**(**rpm\_2**,** 4**,** 0**,** disp\_rpm\_propeller**);**  dtostrf**(**level\_tanki**,** 2**,** 0**,** disp\_tanki**);**  dtostrf**(**steer**,** 3**,** 0**,** disp\_steer**);**  dtostrf**(**depth**,** 2**,** 0**,** disp\_depth**);**  lcd**.**clear**();**    lcd**.**setCursor**(**0**,**0**);** //Start at character 0 on line 0  lcd**.**print**(**"RPM Engine: "**);**  lcd**.**setCursor**(**12**,**0**);** //Start at character 0 on line 0  lcd**.**print**(**disp\_rpm\_engine**);**    lcd**.**setCursor**(**0**,**1**);** //Start at character 0 on line 0  lcd**.**print**(**"RPM Propeller: "**);**  lcd**.**setCursor**(**15**,**1**);**  lcd**.**print**(**disp\_rpm\_propeller**);**    lcd**.**setCursor**(**0**,**2**);** //Start at character 0 on line 0  lcd**.**print**(**"Level Tangki: "**);**  lcd**.**setCursor**(**14**,**2**);**  lcd**.**print**(**disp\_tanki**);**  lcd**.**setCursor**(**17**,**2**);**  lcd**.**print**(**"L"**);**    lcd**.**setCursor**(**0**,**3**);** //Start at character 0 on line 0  lcd**.**print**(**"Steer:"**);**  lcd**.**setCursor**(**6**,**3**);**  lcd**.**print**(**disp\_steer**);**  lcd**.**setCursor**(**9**,**3**);**  lcd**.**print**((**char**)**223**);**    lcd**.**setCursor**(**10**,**3**);** //Start at character 0 on line 0  lcd**.**print**(**" Depth:"**);**  lcd**.**setCursor**(**17**,**3**);**  lcd**.**print**(**disp\_depth**);**  lcd**.**setCursor**(**19**,**3**);**  lcd**.**print**(**"m"**);**  // Delay sebesar 28 tick, 1 tick ~ 18 ms  vTaskDelayUntil**(** **&**xLastWakeTime**,** 10**);**//( 120 / portTICK\_PERIOD\_MS ) );  **}**  **}** |

### Task RPM

|  |
| --- |
| void TaskRPM**(**void **\***pvParameters**)** // This is a task.  **{**  **(**void**)** pvParameters**;**  TickType\_t xLastWakeTime**;**  xLastWakeTime **=** xTaskGetTickCount**();**    int countG**,** countH**;**  **for** **(;;)**  **{**  rpm\_1 **=** speed\_1**.**calcRPM**();**  rpm\_2 **=** speed\_2**.**calcRPM**();**  vTaskDelayUntil**(** **&**xLastWakeTime**,** 1**);**//( 120 / portTICK\_PERIOD\_MS ) );  **}**  **}** |

## Troubleshooting

LED berkedip setiap 4 detik -> overflow

Terjadi segmentation fault -> overflow

Program terkompilasi dan terupload tapi debugging di Serial channel tidak tampak apapun -> kemungkinan besar overflow/memory mikrokontroler tidak cukup