

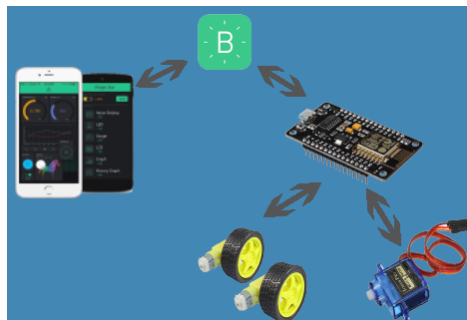
## Bab 3

### PERANCANGAN SISTEM ARM ROBOT

#### 3.1 Perancangan perangkat keras

Cara kerja *Grabber Robot* yang akan dirancang ini adalah sistem kendali sebuah robot lengan oleh mikrokontroler *NodeMCU* dimana data mengenai pergerakannya dikirim dari aplikasi *blynk* yang dikembangkan dengan bahasa pemrograman *arduino*. Aplikasi *blynk* akan mengirim sekaligus menerima data internet ke mikrokontroler *NodeMCU*. Mikrokontroler *NodeMCU* akan memproses data ini untuk mengendalikan robot lengan melalui motor servo sebagai aktuatornya sehingga dapat bergerak sesuai dengan *slider* pengendali yang digeser pada aplikasi *blynk*.

Untuk memahami cara kerja sistem, maka dibuat blok sistem dari perancangan robot yang dibuat. Gambar 3.1 merupakan gambar blok sistem perancangan pengendalian robot lengan 4 DOF menggunakan *blynk*. Sistem ini dibagi menjadi dua bagian yaitu perancangan perangkat keras dan perancangan perangkat lunak.



Gambar 3.1: Skema Kendali Arm Robot

Penjelasan diagram blok pada perancangan perangkat keras dari pengendalian robot lengan 4 DOF dengan aplikasi *blynk* berbasis IoT adalah sebagai berikut :

1. Bagian *Smartphone*

Merupakan perangkat yang digunakan untuk mengirim dan menerima data melalui aplikasi *Blynk*. Bagian ini membutuhkan koneksi internet agar dapat terhubung ke *blynk server*.

2. Bagian Mikrokontroler NodeMCU

Bagian ini berfungsi sebagai pusat kendali dari seluruh sistem yang ada, karena kode program utama terletak pada bagian ini, Merupakan bagian kontroler, semua data-data yang dikirim dari *blynk server* akan dibaca oleh kode program dan disimpan di dalam mikrokontroler NodeMCU. Mikrokontroler yang digunakan adalah mikrokontroler NodeMCU dengan kapasitas *flash memory* sebesar 16MB max (512K normal).

3. Bagian *Arm Robot*

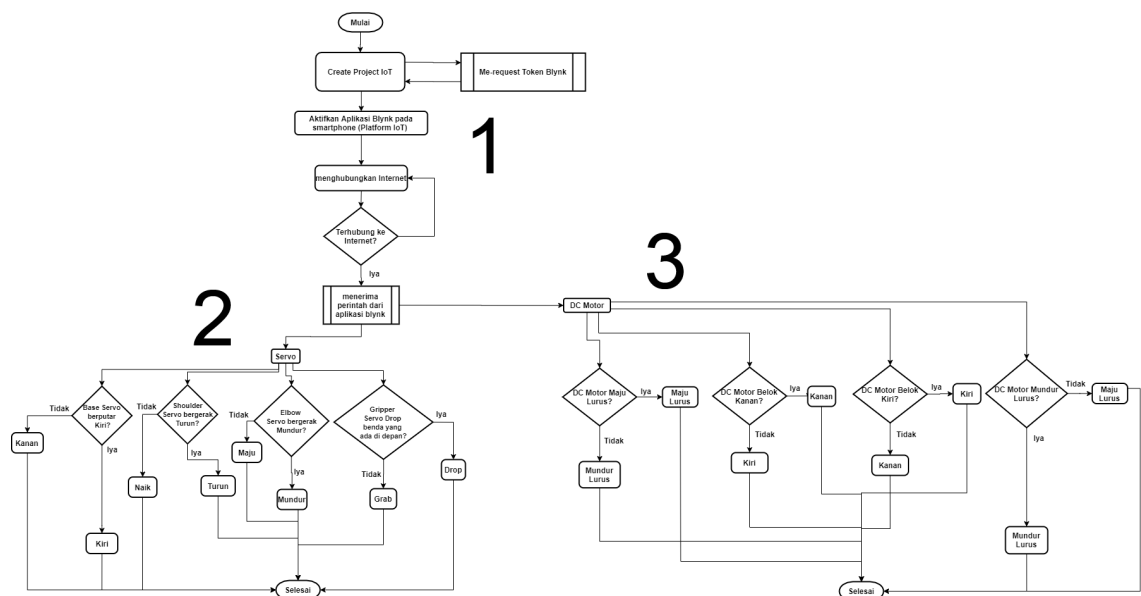
Merupakan *prototype manipulator* mekanik yang terdiri dari empat *joint* dan satu *end-effector*, dimana setiap *joint* dihubungkan oleh *link* yang dicetak dari bahan *acrylic*. Dimana aktuator robot lengan menggunakan motor servo sebagai penggerak sendi-sendi robot yang berada di setiap *joint* dan *end-effector*.

4. Bagian Motor DC

Bagian ini berfungsi sebagai penggerak *Grabber Robot*. DC Motor ini juga dapat disebut sebagai motor arus searah. Seperti namanya, dc motor memiliki dua terminal dan memerlukan tegangan arus searah atau DC (Direct Current) untuk dapat menggerakkannya.

### 3.1.1 Perancangan perangkat lunak

Pembuatan program menggunakan bahasa Arduino. Program dirancang agar mikrokontroler dapat mengirim dan menerima data secara serial sehingga robot lengan dapat bergerak sesuai perintah yang di instruksikan melalui *Blynk*.



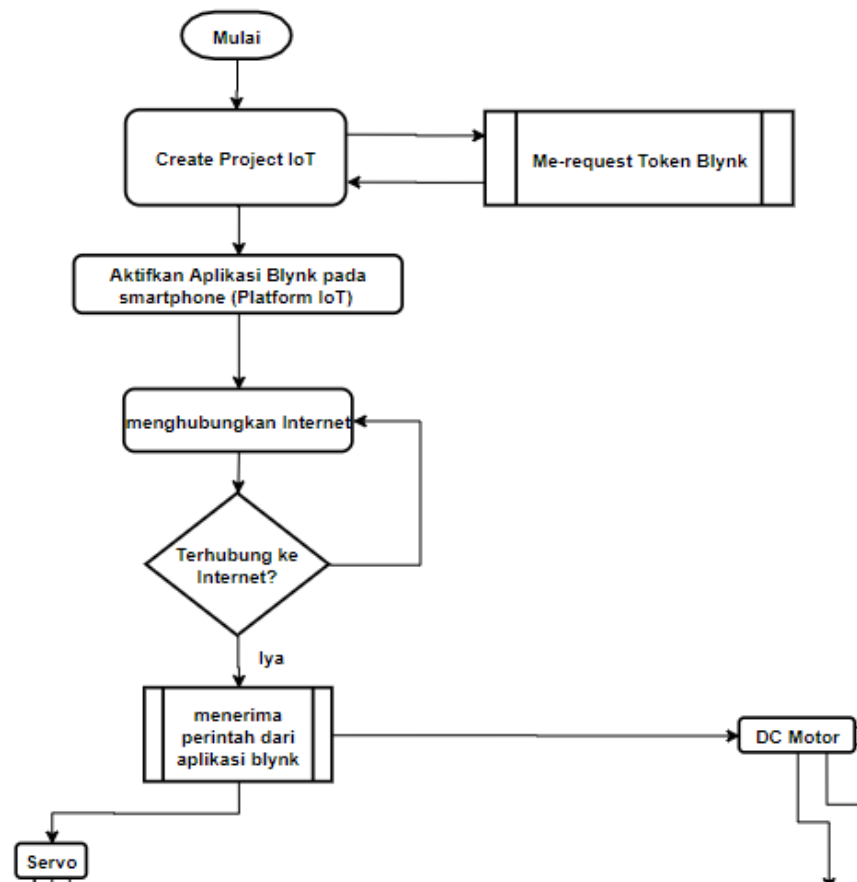
Gambar 3.2: Perancangan perangkat lunak

### 3.2 Analisa Flowchart Secara Detail

Untuk analisa flowchart ini dibagi menjadi 3 bagian diantaranya:

1. Proses pengkoneksian blynk dengan jaringan Wi-Fi pada *smartphone*.

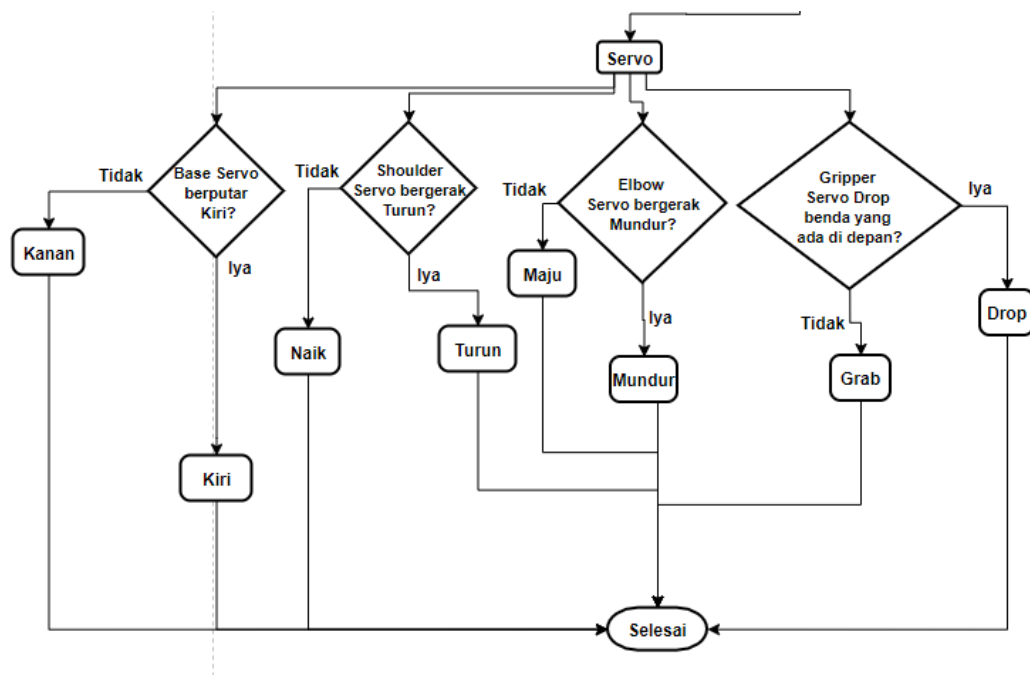
Sebelum membuat sebuah *remote control* pada aplikasi *blynk*, langkah awal ialah dengan *Login/Create new account/Scan QR Code*. jika belum memiliki sebuah akun pada aplikasi *blynk* maka bisa membuat akun baru atau dapat *Login* dengan menggunakan akun *Facebook* jika memiliki akun *Facebook* atau dapat *Scan QR Code* untuk *sharing remote control* tanpa perlu *Login* menggunakan 2 akun, kemudian membuat *remote control* pada aplikasi *blynk*, setelah membuat lakukan pengiriman *token blynk* agar *blynk* dan mikrokontroler dapat dikendalikan, aktifkan aplikasi *blynk* pada *smartphone (platform IoT)*, lalu *blynk* akan menghubungkan ke *internet*, jika tidak terhubung maka lakukan koneksi ulang agar dapat terhubung ke *internet*, jika berhasil terhubung ke *internet* maka *remote control* pada aplikasi *blynk* dapat dikendalikan.



Gambar 3.3: Proses pengkoneksian blynk dengan jaringan Wi-Fi pada *smartphone*.

## 2. Proses penggerakan servo.

- Pada servo menerima perintah dari NodeMCU untuk berputar ke kiri, maka *base servo* akan berputar ke kiri, jika tidak maka *base servo* akan berputar ke arah kanan.
- Pada servo menerima perintah dari NodeMCU untuk bergerak turun, maka *shoulder servo* akan bergerak turun, jika tidak maka *shoulder servo* akan bergerak naik.
- Pada servo menerima perintah dari NodeMCU untuk bergerak mundur, maka *elbow servo* akan bergerak mundur, jika tidak maka *elbow servo* akan bergerak maju.
- Pada servo menerima perintah dari NodeMCU untuk drop benda yang ada di depan, maka *gripper servo* akan drop benda yang ada di depannya, jika tidak maka *gripper servo* akan **grab** benda yang ada di depannya.

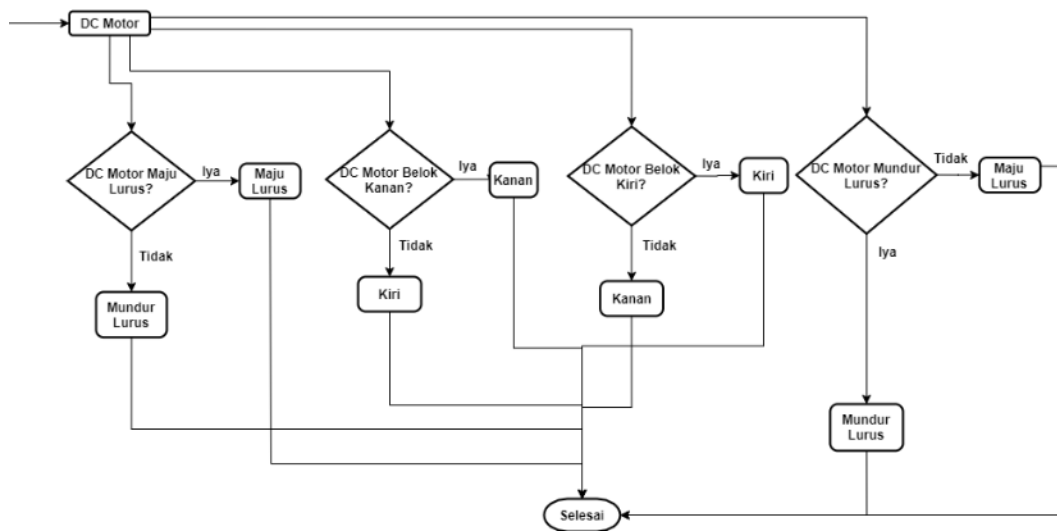


Gambar 3.4: Proses penggerakan motor servo

### 3. Proses penggerakan dc motor.

- Pada motor dc menerima perintah dari NodeMCU untuk bergerak maju lurus, jika iya maka motor dc akan bergerak maju lurus, jika tidak maka motor dc mundur lurus.
- Pada motor dc menerima perintah dari NodeMCU untuk belok kanan, jika iya maka motor dc akan belok kanan, jika tidak maka motor dc akan belok kiri.
- Pada motor dc menerima perintah dari NodeMCU untuk belok kiri, jika iya maka motor dc akan belok kiri, jika tidak maka motor dc akan belok kanan.
- Pada motor dc menerima perintah dari NodeMCU untuk mundur lurus, jika iya maka motor dc akan mundur lurus, jika tidak maka motor dc maju lurus.



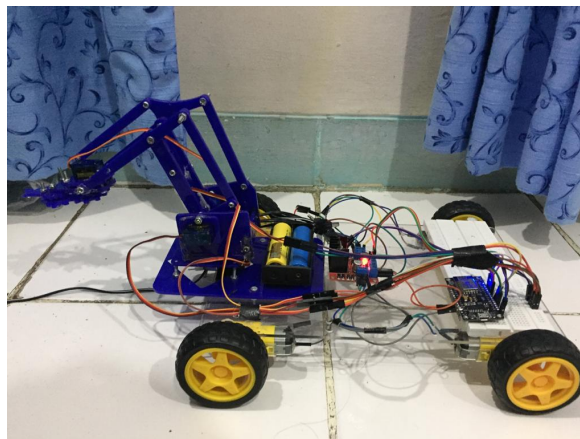


Gambar 3.5: Proses penggerakan dc motor.

### 3.3 Perancangan Mekanik Lengan Robot



Gambar 3.6: Desain mekanik lengan robot



Gambar 3.7: Prototipe lengan robot

Lengan robot dirancang dengan 4 DOF (*Degree Of Freedom*) yang terdiri dari 4 buah servo dengan konfigurasi sebagai berikut:

1. DOF 1 (Servo1): *Base Servo*
2. DOF 2 (Servo2): *Elbow Servo*
3. DOF 3 (Servo3): *Shoulder Servo*
4. DOF 4 (Servo4): *Gripper Servo*

*Arm Robot* ini memiliki spesifikasi mekanik dengan ketinggian 15cm, panjang lengan robot 25cm, lebar alas 14.1cm, panjang alas 9.2cm. *Base Servo* digunakan untuk memutar sebesar 180 derajat. *Elbow Servo* digunakan untuk menaikkan atau menurunkan. *Shoulder Servo* digunakan untuk maju dan mundur. *Gripper Servo* digunakan untuk mengambil dan melepas barang yang ada di depannya.

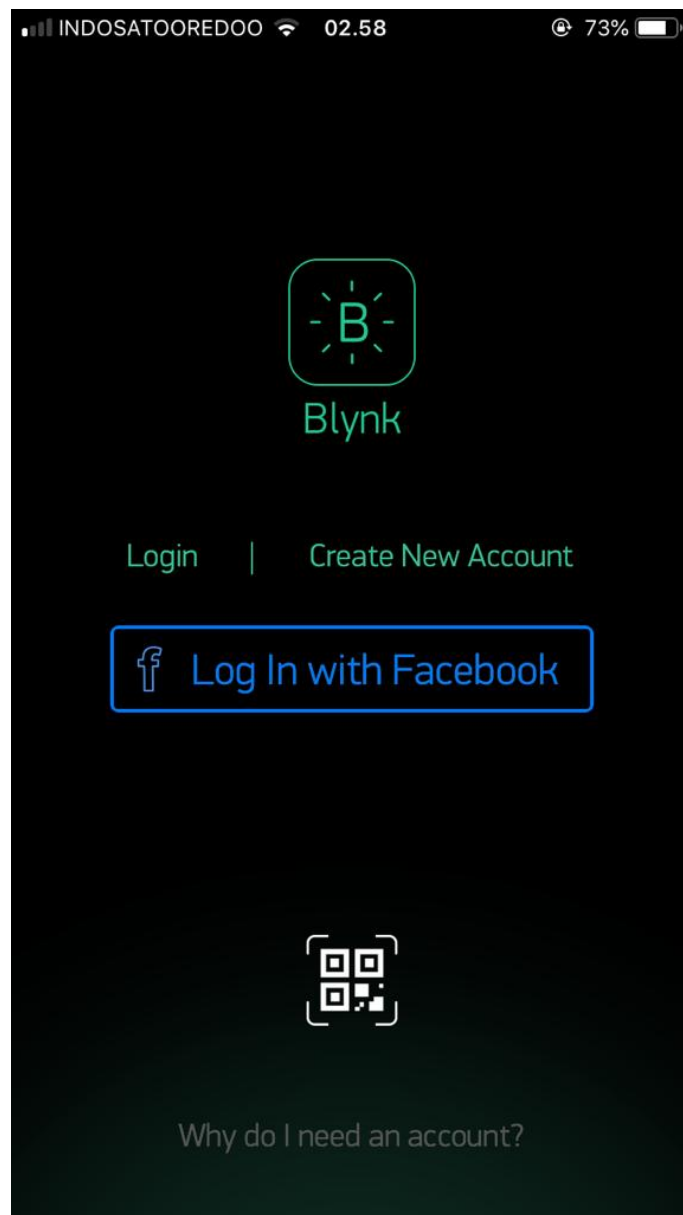
### 3.3.1 Pembuatan Remote Control pada Blynk

1. Download aplikasi blynk pada Playstore/AppStore
2. *Install blynk library.*

Jika memiliki sebuah NodeMcu atau mikrokontroler yang dapat terhubung ke internet maka untuk menggerakkan programnya dibutuhkan sebuah library. Hal ini di haruskan karena untuk memprogram pada sebuah arduino IDE diwajibkan memiliki library, agar program dapat berjalan.

3. Tampilan awal blynk

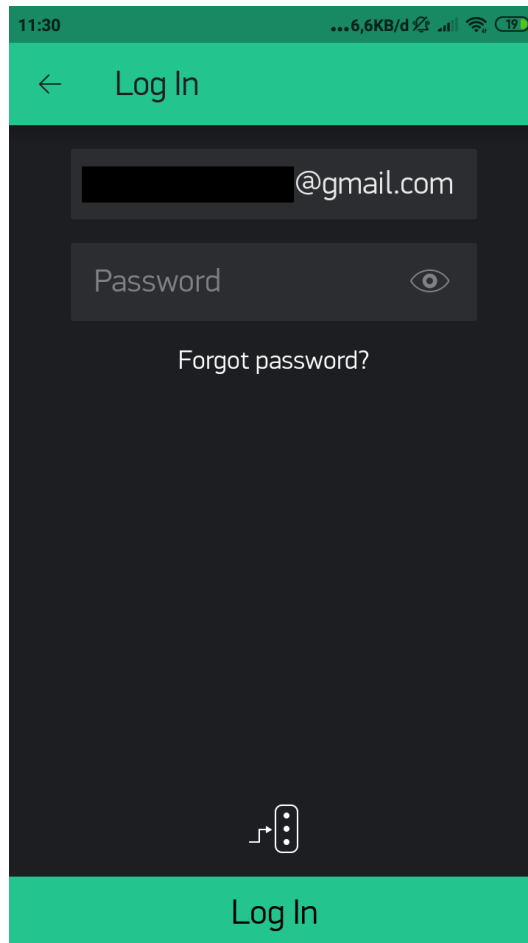
Sebelum membuat sebuah remote control pada aplikasi Blynk, langkah awal ialah dengan *Login/Create New Account/Scan QR Code*. Jika belum memiliki sebuah akun pada aplikasi Blynk maka bisa membuat akun baru atau dapat LogIn dengan menggunakan akun Facebook jika memiliki akun Facebook atau dapat *Scan QR Code* untuk *Sharing Remote Control* tanpa perlu *LogIn* menggunakan 2 akun.



Gambar 3.8: Tampilan awal blynk

#### 4. Tampilan Log In

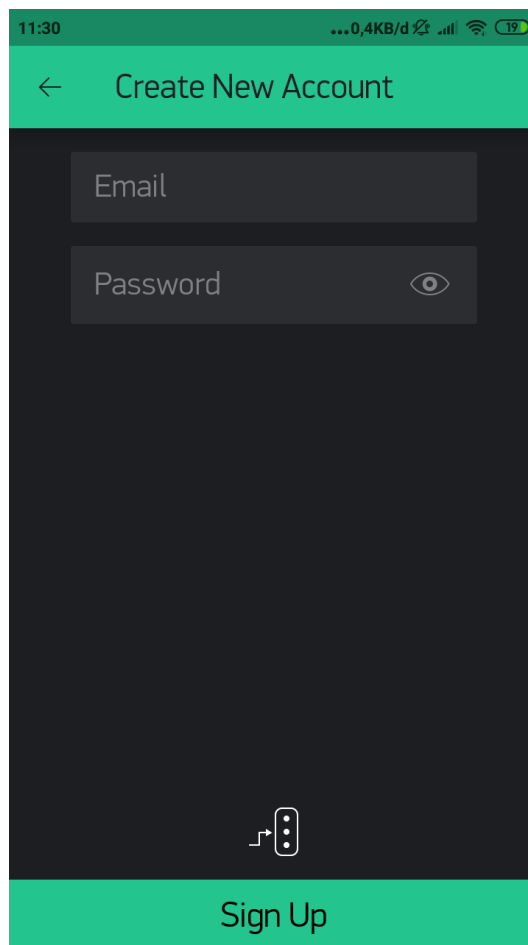
Jika sudah memiliki akun pada blynk dan tidak ingin LogIn dengan menggunakan facebook, maka dapat LogIn dengan menggunakan akun blynk.



Gambar 3.9: Tampilan Log In

## 5. Tampilan membuat akun baru

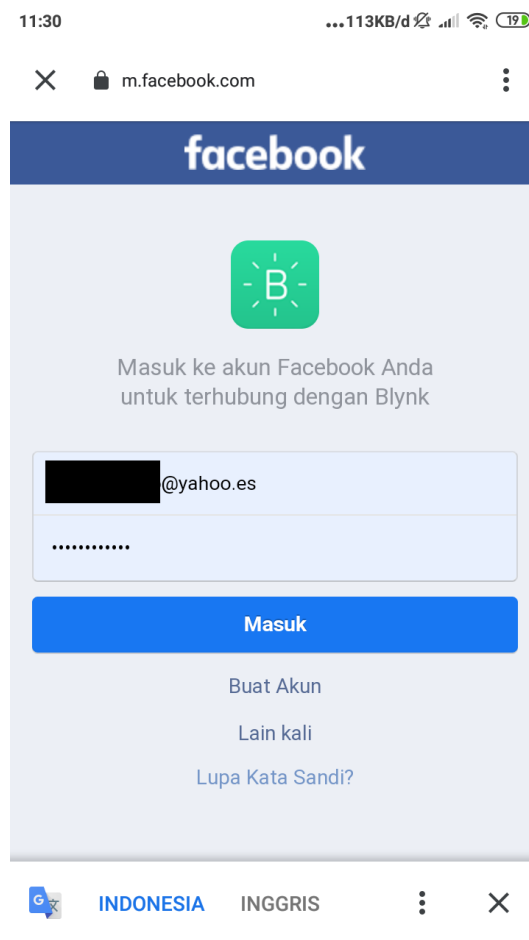
Tampilan ini adalah untuk membuat akun baru. Untuk membuat akun baru pada *blynk*, masukkan email dan password yang diinginkan, setelah itu pilih *Sign Up*, setelah *Sign Up* periksa email untuk mendapatkan bahwa akun telah terdaftar dan mendapatkan kode autentikasi.



Gambar 3.10: Tampilan membuat akun baru

#### 6. Tampilan untuk LogIn dengan menggunakan Facebook

Jika tidak ingin membuat sebuah akun pada blynk, maka dapat *LogIn* dengan menggunakan akun Facebook. Setelah *LogIn* periksa email untuk mendapatkan bahwa akun telah terdaftar dan mendapatkan kode autentikasi.

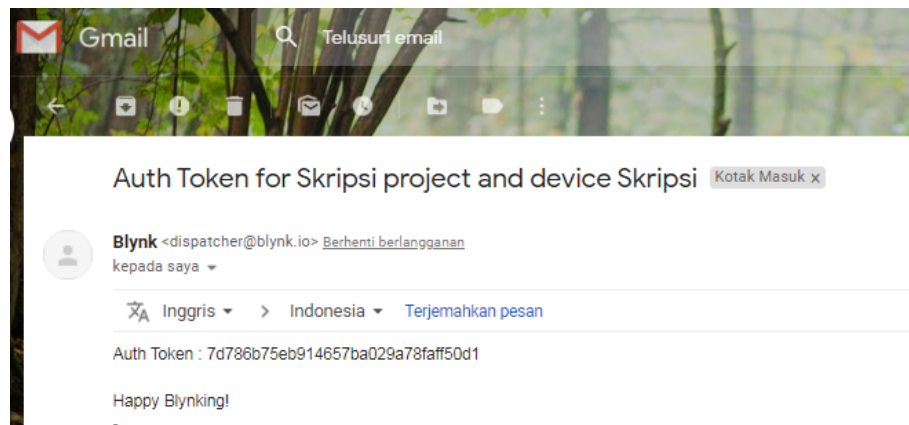


Gambar 3.11: Tampilan Log In menggunakan Facebook



## 7. Tampilan kode autentikasi pada halaman gmail

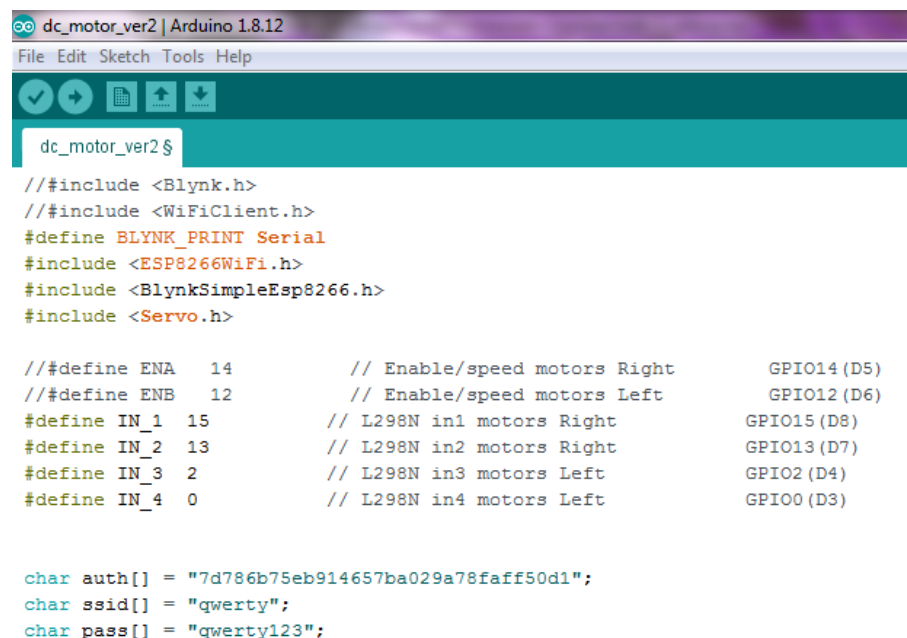
Jika sudah membuat sebuah akun pada blynk atau *LogIn* dengan menggunakan Facebook, maka akan dapat kode autentikasi pada gmail yang di gunakan. Kode ini berfungsi sebagai penghubung antara blynk server dengan mikrokontroler yang digunakan. Setelah mendapatkan kode autentikasi maka *copy-paste* ke dalam arduino IDE.



Gambar 3.12: Tampilan kode autentikasi pada halaman gmail

## 8. Tampilan arduino IDE

Jika sudah mendapatkan kode autentikasi pada email yang telah di daftarkan pada akun blynk, maka copy-paste pada arduino IDE. Untuk memprogram sebuah akun blynk yang berbasis IoT (*Internet of Things*), ada hal yang wajib di *input* ke dalam *sketch* arduino IDE, yaitu berupa *library blynk* (# define BLYNK\_ PRINT Serial), (# include (BlynkSimpleEsp8266.h)), dan char auth, ssid, pass. Tanpa *library* dan *char* yang telah disebutkan, maka blynk tidak dapat terhubung ke mikrokontroler.



```

dc_motor_ver2 | Arduino 1.8.12
File Edit Sketch Tools Help

dc_motor_ver2 $

//#include <Blynk.h>
//#include <WiFiClient.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Servo.h>

//#define ENA 14          // Enable/speed motors Right      GPIO14 (D5)
//#define ENB 12          // Enable/speed motors Left       GPIO12 (D6)
#define IN_1 15           // L298N in1 motors Right  GPIO15 (D8)
#define IN_2 13           // L298N in2 motors Right  GPIO13 (D7)
#define IN_3 2            // L298N in3 motors Left   GPIO2 (D4)
#define IN_4 0            // L298N in4 motors Left   GPIO0 (D3)

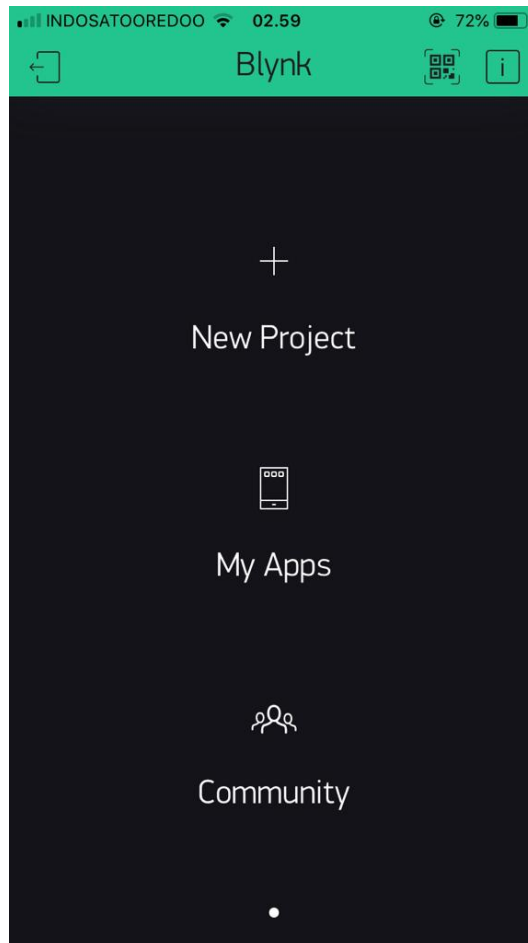
char auth[] = "7d786b75eb914657ba029a78faff50d1";
char ssid[] = "qwerty";
char pass[] = "qwerty123";

```

Gambar 3.13: Tampilan Arduino IDE

## 9. Tampilan kedua blynk

Pada tampilan kedua terdapat *New Project* (untuk membuat project baru), *My Apps* (jika sudah memiliki project), *Community* (sebuah komunitas atau sebuah forum untuk tanya jawab).



Gambar 3.14: Tampilan kedua blynk

#### 10. Tampilan Membuat *Project* Baru.

Untuk membuat sebuah *project* pada aplikasi Blynk ialah dengan cara:

- I Masukkan nama *project* yang diinginkan
- II Pilih Jenis *device* atau jenis mikrokontroler yang digunakan.
- III Setelah itu pilih *Connection Type*, disini apakah menggunakan *Internet, Bluetooth, GSM, WiFi* atau *USB*. Untuk lebih jelasnya lihat pada gambar 3.16.
- IV Pilih jenis Tampilan pada aplikasi blynk apakah ingin menggunakan gelap atau terang.

INDOSATOORED00 03.00 71%

× New Project

Skripsi I

CHOOSE DEVICE

NodeMCU II ↓

CONNECTION TYPE

WiFi III ↓

THEME

DARK ☒ LIGHT IV

Create Project

Gambar 3.15: Membuat project baru

## 11. Memilih jenis koneksi

Untuk pada project ini, mikrokontroler yang di gunakan yaitu menggunakan NodeMCU, dan jenis koneksi yang di sediakan oleh aplikasi blynk berupa:

- *GSM*

*GSM* merupakan singkatan dari *Global System for Mobile Communications*. Jaringan *GSM* bisa diartikan sebagai sebuah teknologi komunikasi seluler yang bersifat *digital*. Teknologi *GSM* banyak diterapkan pada komunikasi bergerak, khususnya telepon genggam.

- *USB*

*USB* adalah singkatan dari *Universal Serial Bus* dan merupakan media penghubung antara komputer dengan perangkat-perangkat elektronik lainnya seperti mouse, keyboard, printer, scanner, ponsel, *flash drive*, *DVD writer*, Konsol permainan, kamera, modem dan bahkan digunakan sebagai media penghubung untuk mengendalikan alat-alat uji dan mesin-mesin produksi.

- *Wi-Fi*

"*Wireless Fidelity*" atau disingkat *Wi-Fi* adalah suatu teknologi yang memakai gelombang radio untuk menghubungkan perangkat (PC, laptop, smartphone) ke jaringan komputer. Atau definisi *Wi-Fi* yaitu teknologi yang menggunakan gelombang radio supaya komputer bisa mengakses internet.

- *Bluetooth*

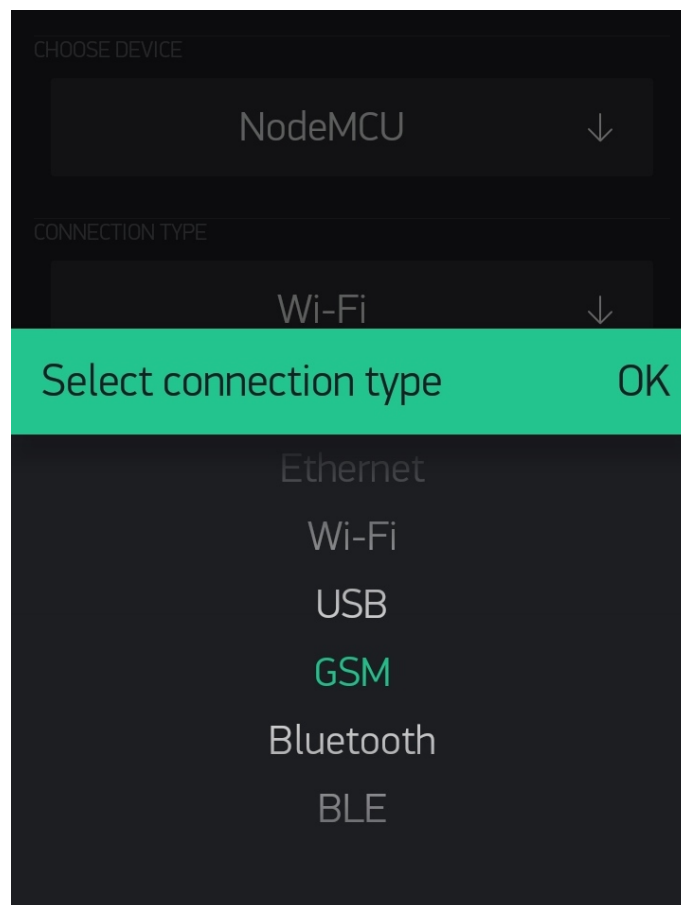
*Bluetooth* merupakan *chip radio* yang dimasukkan ke dalam komputer, printer, handphone dan sebagainya. *Chip bluetooth* ini dirancang untuk menggantikan kabel.

- *BLE*

*Bluetooth Low Energy*(*BLE*) merupakan perkembangan dari *Classic Bluetooth* yang khusus di implementasikan pada teknologi *Internet of Things* (*IoT*).

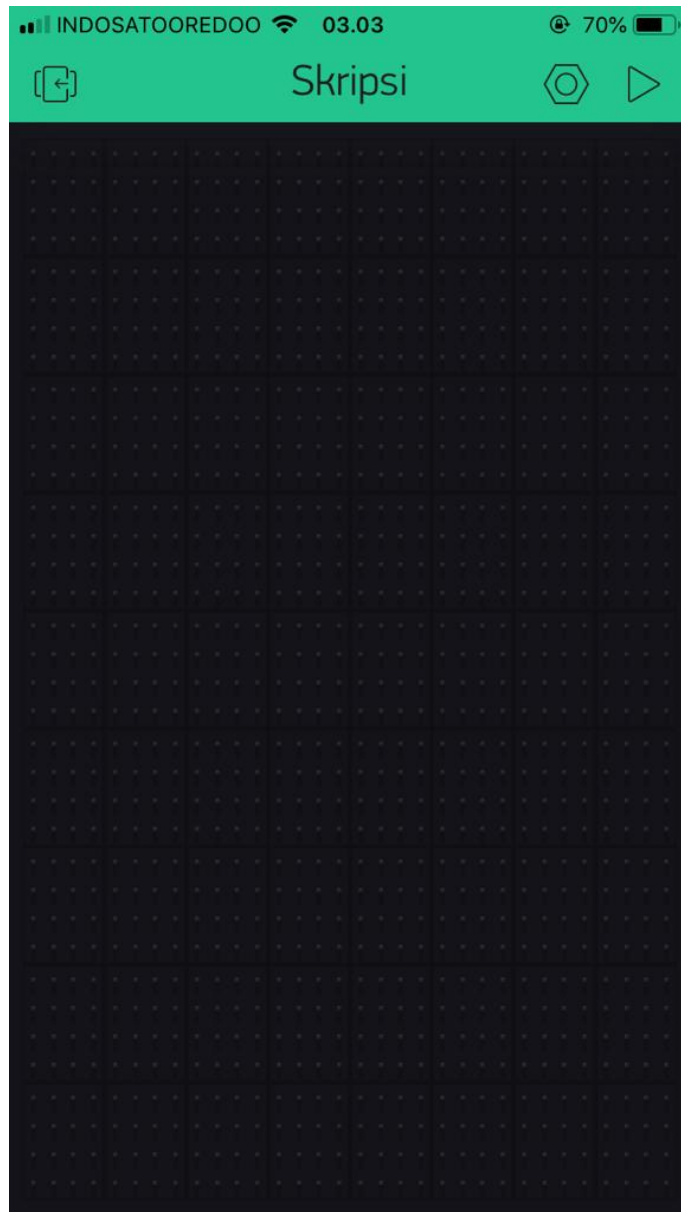
- *Ethernet*

Ethernet merupakan teknologi jaringan yang paling terkenal dan paling banyak digunakan di dunia saat ini. Dengan *Ethernet* kita bisa menghubungkan banyak komputer dengan biaya rendah dan fleksibilitas tinggi.



Gambar 3.16: Jenis koneksi

## 12. lembar project

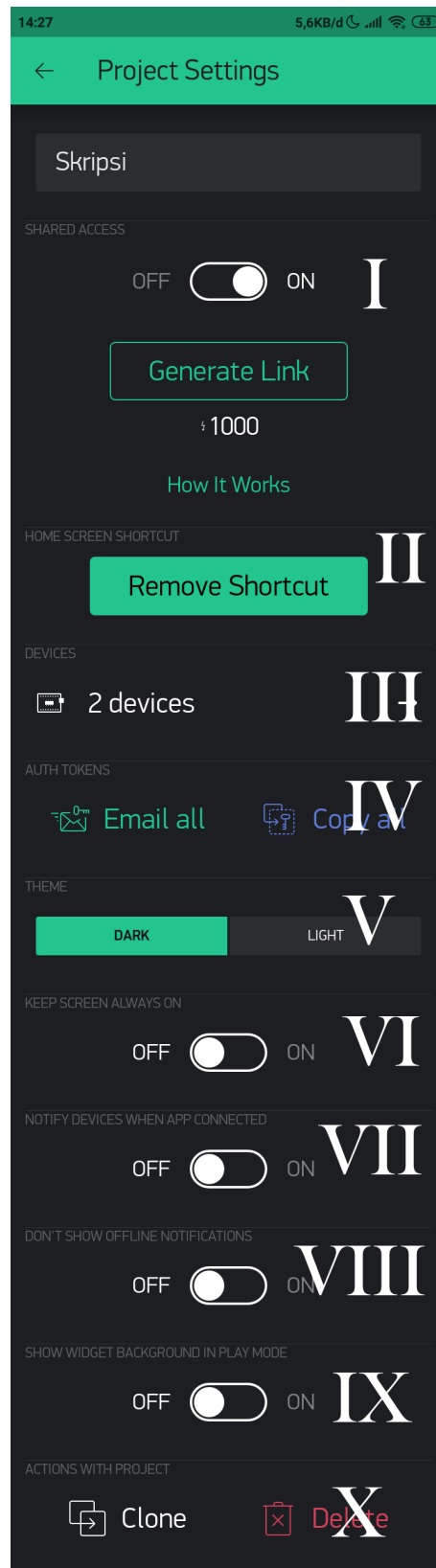
Gambar 3.17: lembar *project*



### 13. *Project Settings*

Pada bagian ini digunakan untuk mengatur *project* dan *sharing access remote control* dimana *project* yang dibuat tidak dapat dimodifikasi selain admin yang membuat *project* tersebut. Berikut penjelasan beberapa bagian yang terdapat pada *Project Settings*:

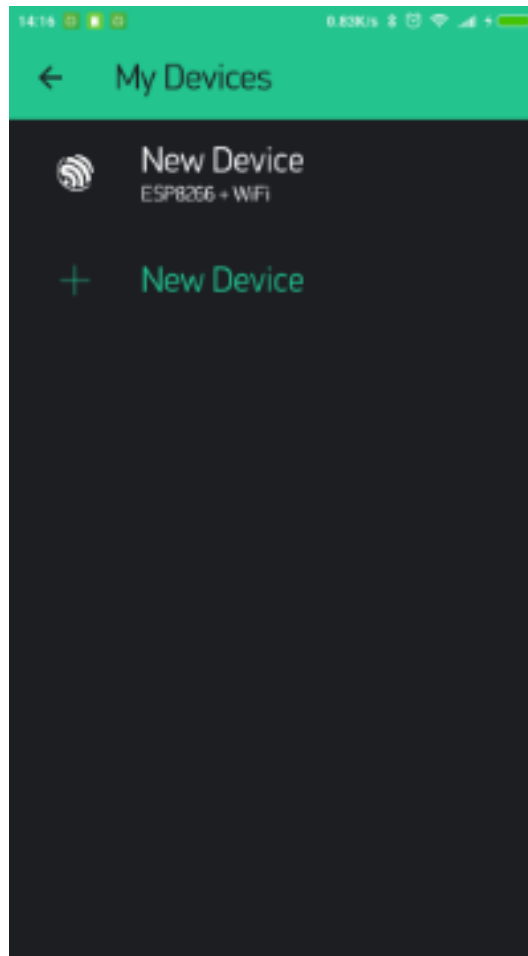
- I *Sharing access remote control* hanya bisa dilihat dan digunakan oleh *user* tetapi *user* tidak mendapatkan hak akses untuk mengatur dan merubah *project* tersebut, dengan catatan *project* yang di *shared* harus memiliki 1000 *energy* seperti pada gambar 3.21.
- II Menambahkan *shortcut* pada *home screen*.
- III Dapat menambahkan *device* baru dengan cara *click Devices* lalu pilih *New Devices*. Untuk lebih jelasnya lihat pada gambar 3.18.
- IV Untuk dapat mengendalikan sebuah *remote control* pada blynk yang telah di buat, maka harus mengirim sebuah *AUTHENTICATION TOKEN* ke sebuah email yang telah digunakan pada akun blynk ini.
- V Mengubah tampilan menjadi gelap atau terang.
- VI *Keep screen always on*.
- VII Memberi notifikasi pada *device*, jika sudah terkoneksi dengan Wi-Fi.
- VIII Tidak memberikan notifikasi, jika blynk *offline*.
- IX Menampilkan pada *background*, jika blynk sedang berjalan.
- X Dapat *sharing project* tanpa perlu melalui admin, tetapi dengan syarat admin sudah memberikan hak akses kepada *user* agar *user* mendapatkan *cloning* dari admin, dengan cara *click* pada bagian *Clone* lalu *user scan QR Code* pada *smartphone* admin.



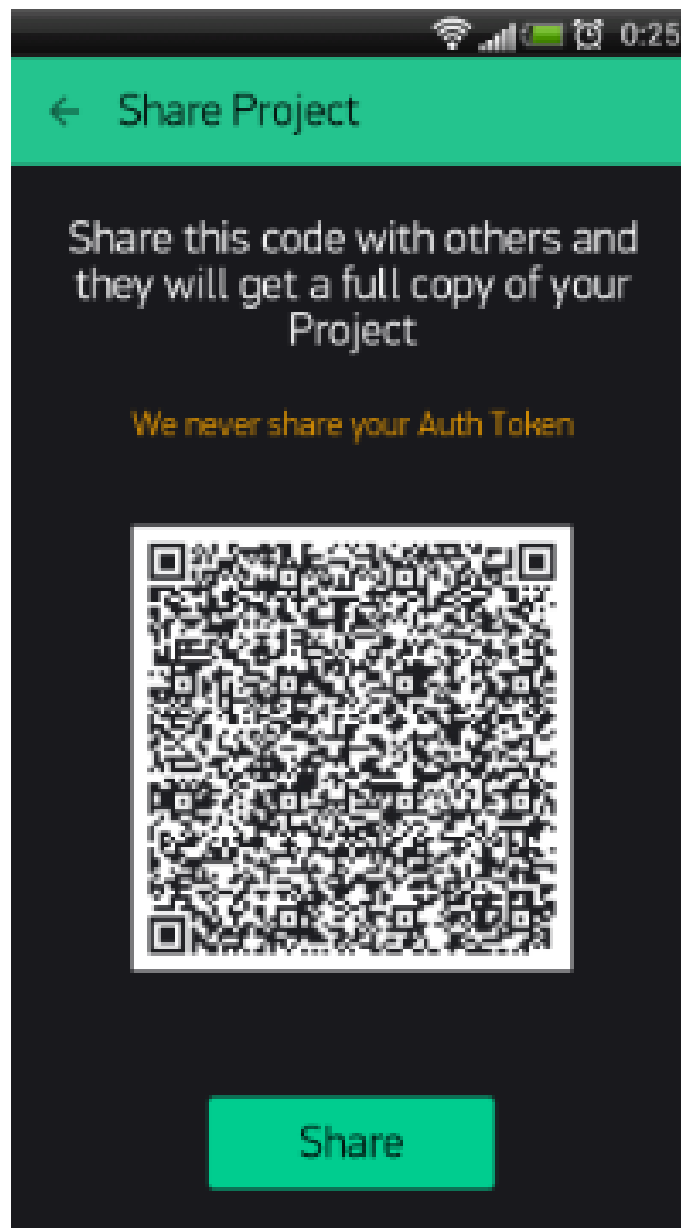
Gambar 3.18: *Project Settings*

#### 14. Menambah *devices* baru

Pada bagian ini, *device* baru dapat di tambahkan sebanyak yang di inginkan, dan terdapat berbagai jenis mikrokontroler yang dapat di pilih seperti, NodeMCU, Arduino Uno, dll..



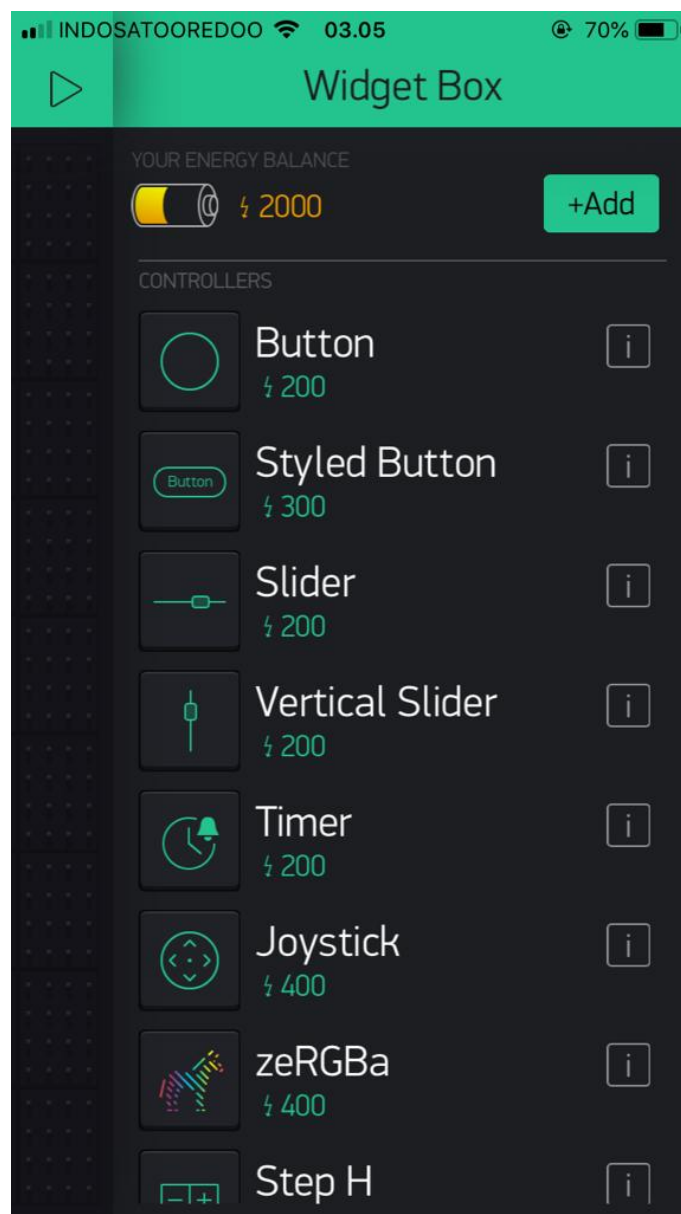
Gambar 3.19: Menambah *devices* baru



Gambar 3.20: *Share Project*

### 15. *Widget Box*

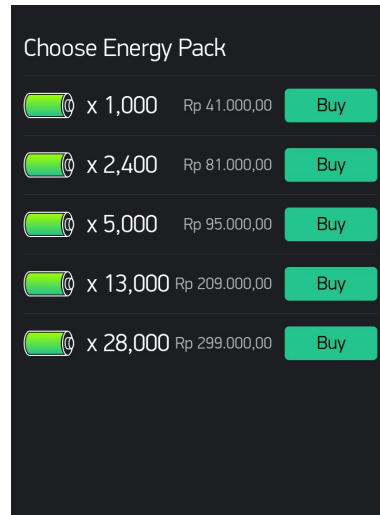
Di bagian *Widget Box* ini terdapat banyak *tools* yang digunakan untuk membuat sebuah *project* yang berbasis IoT.



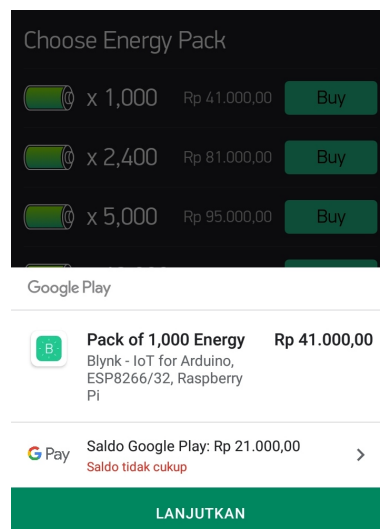
Gambar 3.21: *Widget Box*

## 16. *Energy* pada *widget box*

Pada bagian ini *energy* bawaan blynk sejumlah 2.000, namun untuk menambah *energy* hingga lebih banyak, di haruskan melakukan pembelian dan pembayaran dengan menggunakan *google play/iphone store*.



Gambar 3.22: Memilih *energy* untuk di beli

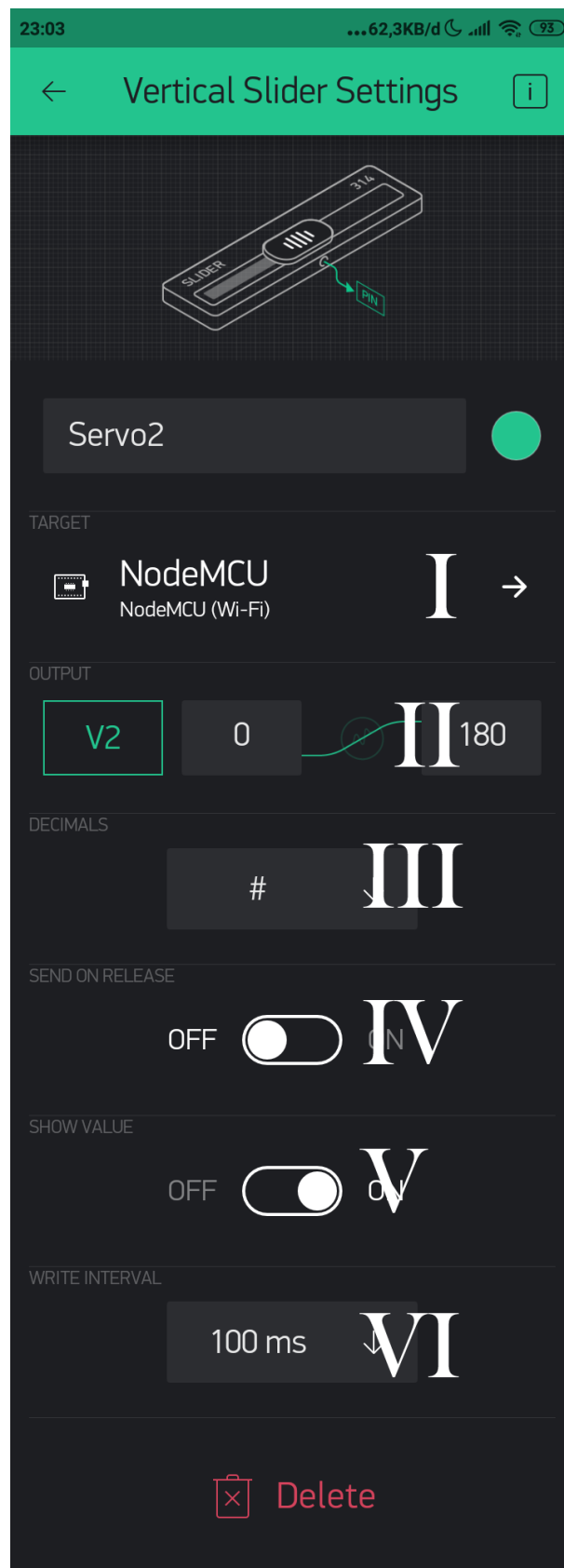


Gambar 3.23: Melakukan pembayaran menggunakan *google play*

### 17. *Vertical Slider Settings*

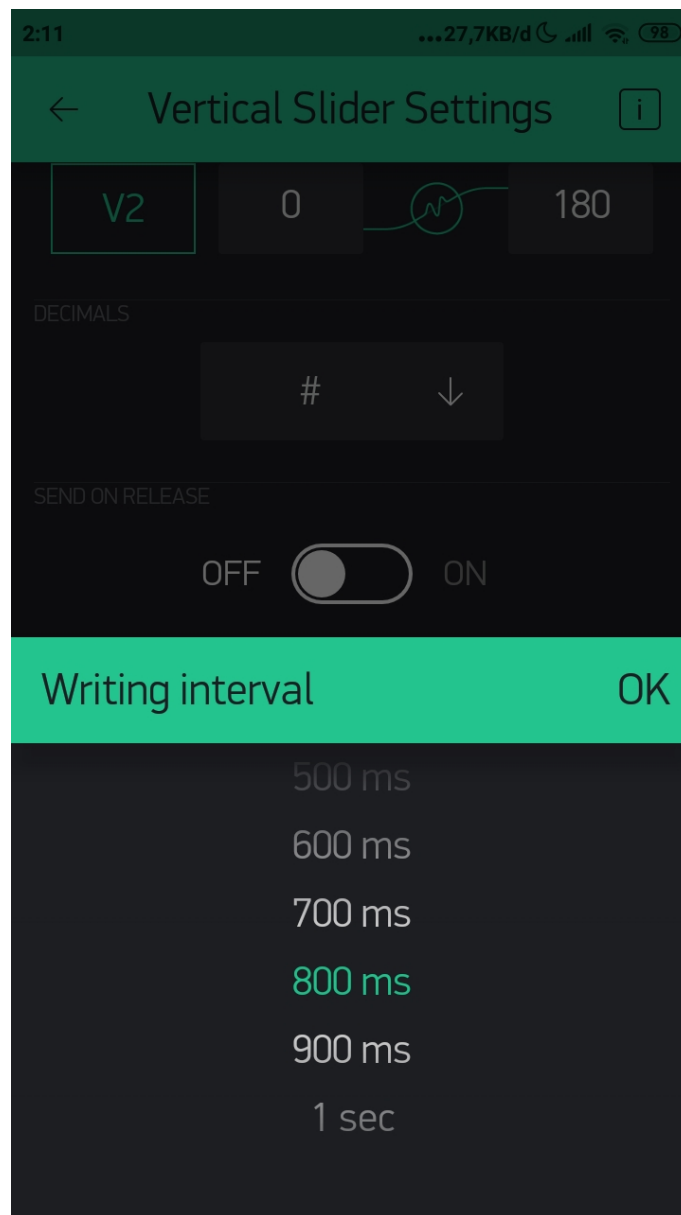
Pada tampilan ini merupakan tampilan untuk mengatur slider pada *blynk remote control*. Fungsinya sama seperti *horizontal slider*. Perbedaanannya ialah pada posisi slidernya. Kalau untuk *slider* ini posisinya dalam keadaan *vertical*, sedangkan *horizontal slider* dalam keadaan posisi *horizontal*. Slider ini di fungsikan untuk menggerakkan jenis *Shoulder servo* dan *Elbow servo*. Berikut beberapa bagian yang terdapat pada bagian *slider settings*:

- I Jenis mikrokontroler yang di gunakan.
- II Jenis pin yang di gunakan, yaitu virtual pin.
- III *Decimals* ini di gunakan jika nilai outputnya bernilai *decimals*.
- IV Pada bagian ini, jika di aktifkan, maka perubahan *output* tidak langsung terjadi saat slider digerakkan akan tetapi, terjadi saat jari melepas perubahan slider. Namun, jika tidak di aktifkan maka perubahan *output* akan langsung terjadi saat slider digerakkan.
- V Pada bagian ini, jika di aktifkan maka tampilan *vertical slider* akan menampilkan nilai derajat servo, namun jika tidak di aktifkan, maka nilai derajat servo tidak akan tampil di *vertical slider*.
- VI Pada bagian ini, di gunakan untuk mengirimkan sinyal kecepatan dari blynk menuju servo. Semakin kecil nilai *wirting interval* (100 ms) maka akan semakin cepat proses pengiriman sinyal dari blynk ke servo, namun jika semakin besar nilai *writing interval* maka akan semakin lambat dalam proses pengiriman sinyal dari blynk ke servo. Untuk lebih jelasnya bisa di lihat pada gambar 3.25.



Gambar 3.24: *Vertical slider settings*



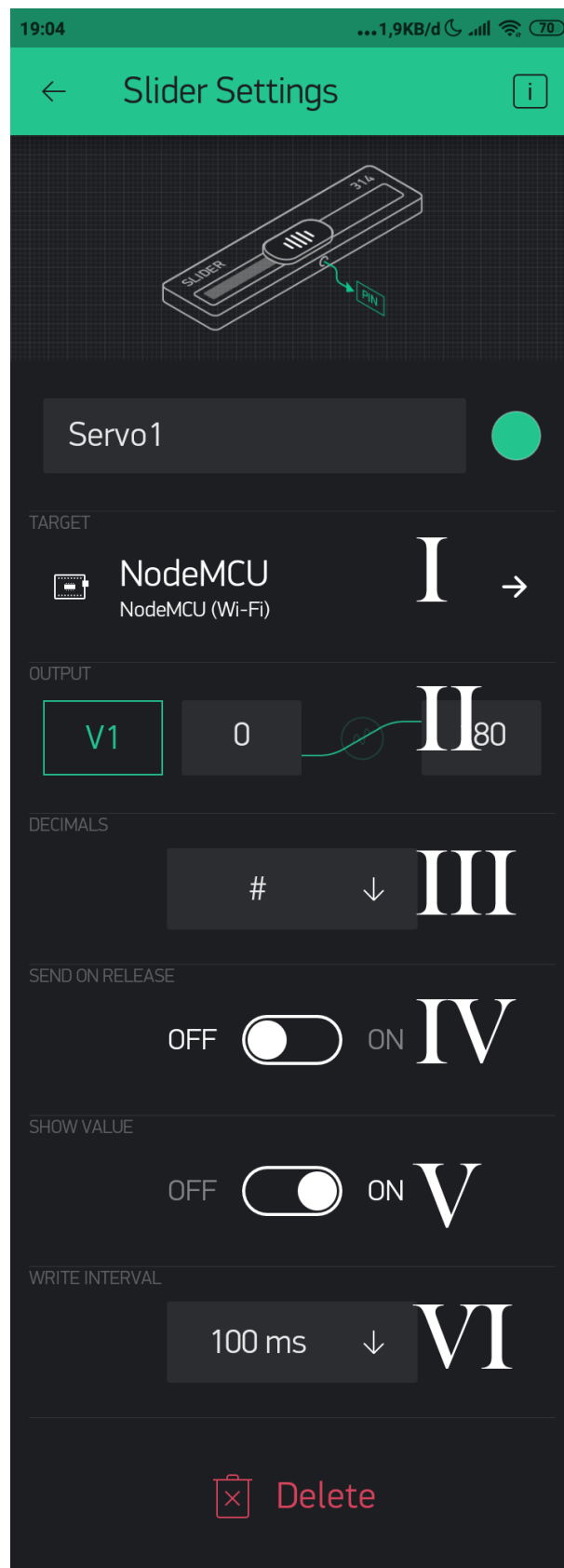


Gambar 3.25: *Writing interval*

## 18. *Slider Settings*

Pada tampilan ini merupakan tampilan untuk mengatur *slider* pada *blynk remote control*. Fungsinya sama seperti *Vertical Slider*, perbedaannya ialah pada posisi slidernya. Kalau untuk *slider* ini posisinya dalam keadaan *horizontal*, sedangkan *vertical slider* dalam keadaan posisi *vertical*. Slider ini di fungsikan untuk menggerakkan jenis *base servo*. berikut beberapa bagian yang terdapat pada *slider settings*:

- I Jenis mikrokontroler yang di gunakan.
- II Jenis pin yang di gunakan, yaitu virtual pin.
- III *Decimals* ini di gunakan jika nilai outputnya bernilai *decimals*.
- IV Pada bagian ini, jika di aktifkan, maka perubahan *output* tidak langsung terjadi saat slider digerakkan akan tetapi, terjadi saat jari melepas perubahan slider. Namun, jika tidak di aktifkan maka perubahan *output* akan langsung terjadi saat slider digerakkan.
- V Pada bagian ini, jika di aktifkan maka tampilan *vertical slider* akan menampilkan nilai derajat servo, namun jika tidak di aktifkan, maka nilai derajat servo tidak akan tampil di *vertical slider*.
- VI Pada bagian ini, di gunakan untuk mengirimkan sinyal kecepatan dari blynk menuju servo. Semakin kecil nilai *wirting interval* (100 ms) maka akan semakin cepat proses pengiriman sinyal dari blynk ke servo, namun jika semakin besar nilai *writing interval* maka akan semakin lambat dalam proses pengiriman sinyal dari blynk ke servo. Untuk lebih jelasnya bisa di lihat pada gambar 3.25.

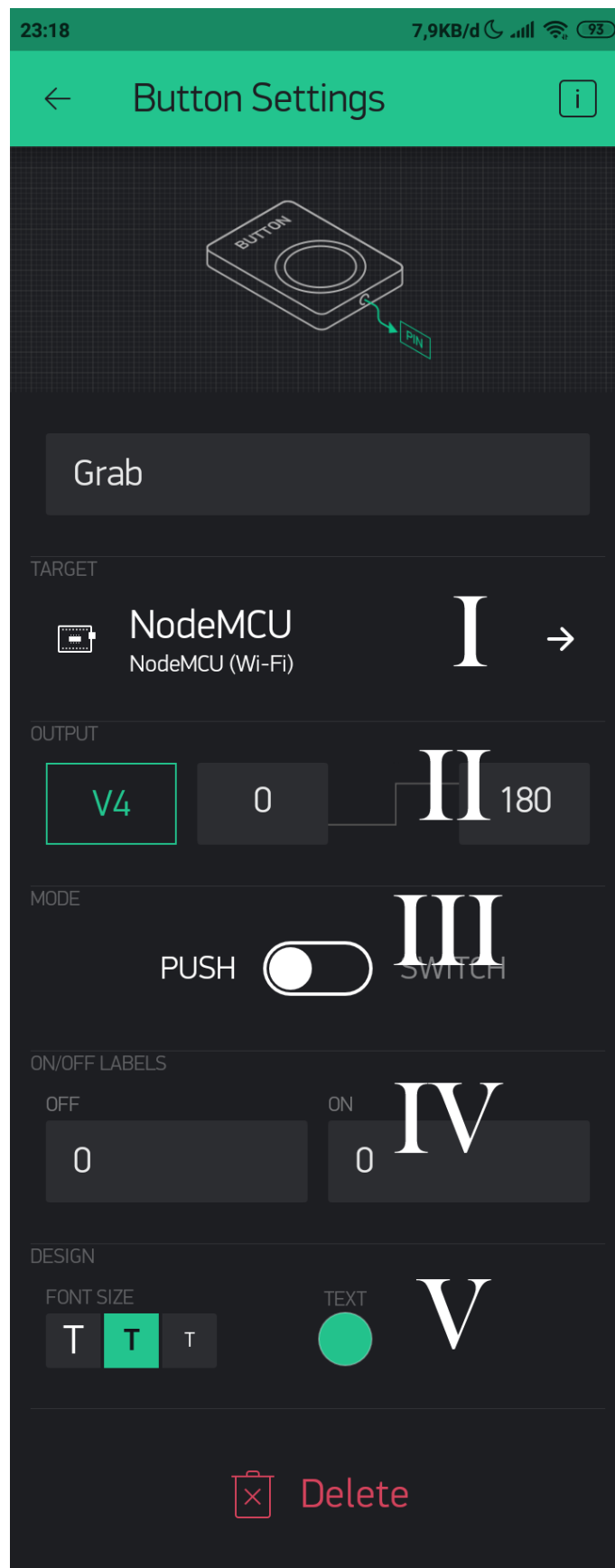


Gambar 3.26: *Horizontal Slider Settings*

## 19. Button Settings

Pada tampilan ini merupakan untuk mengatur sebuah tombol. Dalam *project* ini terdapat 2 *button*, *button* pertama di gunakan untuk *Grab*, maka servo akan bergerak dari 180 derajat menuju 0 derajat, sedangkan *button* ke dua di gunakan untuk *drop*, maka servo akan bergerak dari 0 derajat menuju 180 derajat. Berikut penjelasan beberapa bagian yang terdapat pada *Button Settings*:

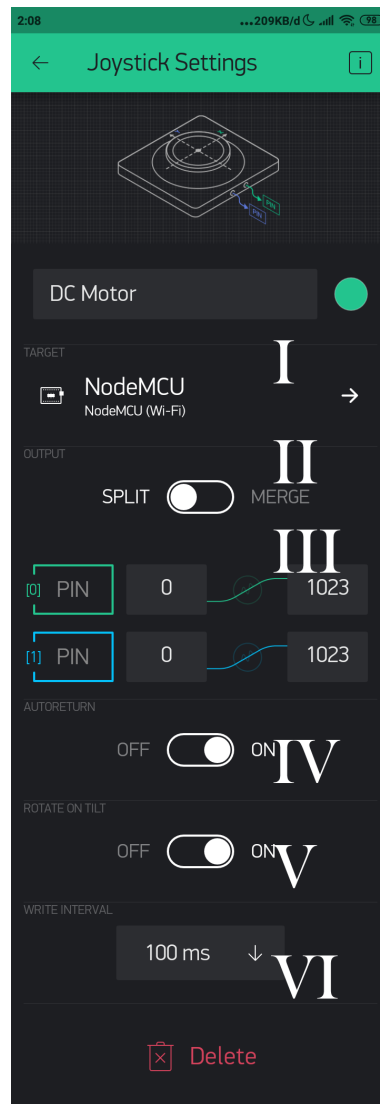
- I Jenis mikrokontroler yang di gunakan.
- II Jenis pin yang di gunakan, yaitu virtual pin.
- III Pada *button* ini terdapat 2 mode, yaitu mode *switch* dan mode *push*. Jika menggunakan mode *push* ketika *button* tersebut di tekan lalu lepas, maka akan kembali ke posisi awal. Sedangkan jika menggunakan mode *switch*, ketika *button* tersebut di tekan lalu lepas, maka akan tetap di posisi akhir.
- IV Pada bagian ini merupakan untuk pemberian label nama *ON* dan *OFF* pada *button*.
- V Pada bagian ini di gunakan untuk merubah ukuran text menjadi kecil, sedang ataupun besar, dan dapat merubah warna text.

Gambar 3.27: *Button Settings*

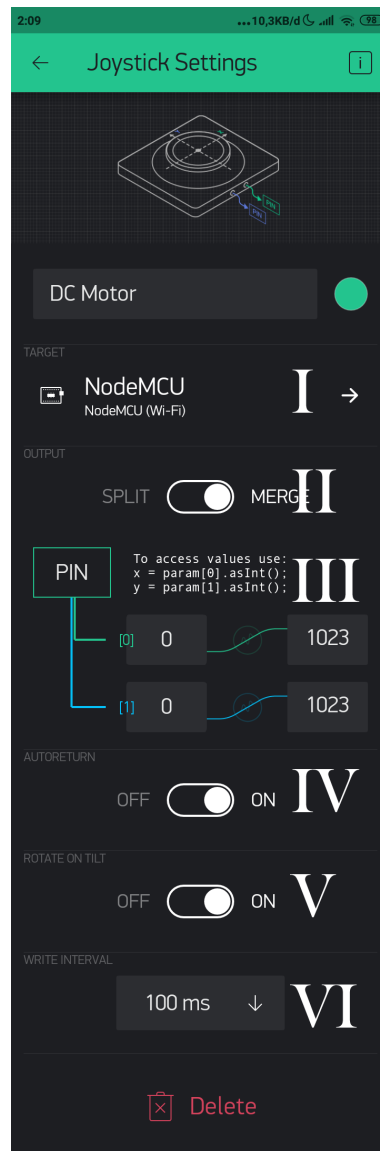
## 20. Joystick Settings

Pada bagian ini digunakan untuk menggerakkan sebuah *DC Motor* dengan menggunakan Joystick. Berikut penjelasan beberapa bagian yang terdapat pada *Joystick settings*:

- I Jenis mikrokontroler yang di gunakan.
- II Pada bagian ini joystick dapat menggunakan mode *SPLIT* atau *MERGE*. Jika menggunakan mode *split* maka terdapat 2 pin yang dapat digunakan. sedangkan jika menggunakan mode *merge* maka hanya terdapat 1 pin yang dapat digunakan. Untuk lebih jelasnya bisa di lihat pada gambar 3.28 dan 3.29.
- III Pada bagian ini ketika *auto return* di aktifkan, maka joystick yang telah di gerakan akan kembali ke posisi awal yaitu di tengah, sedangkan jika *auto return* tidak di aktifkan, maka joystick yang telah di gerakan tidak akan kembali ke posisi awal.
- IV *Rotate on tilt*
- V Pada bagian ini, di gunakan untuk mengirimkan sinyal kecepatan dari blynk menuju servo. Semakin kecil nilai *wirting interval* (100 ms) maka akan semakin cepat proses pengiriman sinyal dari blynk ke servo, namun jika semakin besar nilai *writing interval* maka akan semakin lambat dalam proses pengiriman sinyal dari blynk ke servo.



Gambar 3.28: Joystick Settings dengan mode split



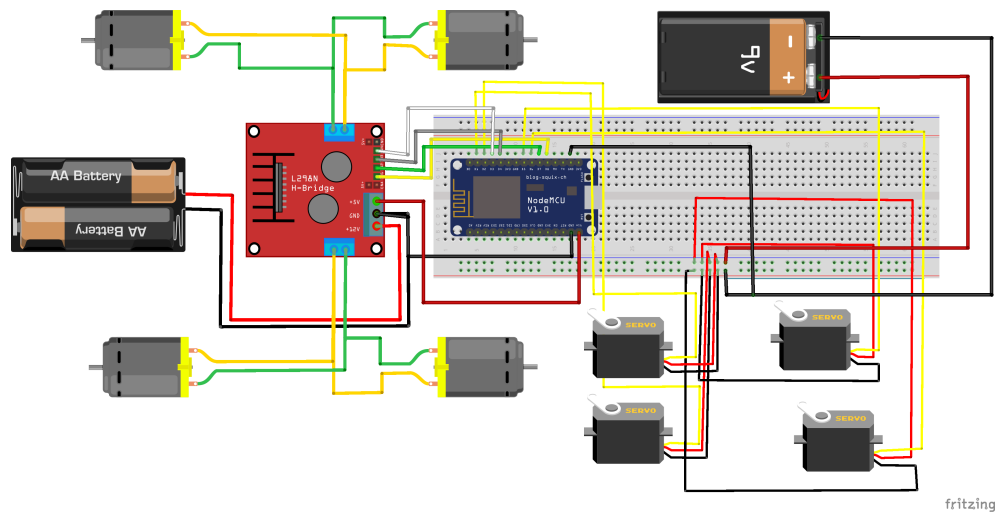
Gambar 3.29: Joystick Settings dengan mode merge



### 3.4 Analisa Rangkaian Secara Detail

Input untuk rangkaian robot lengan adalah blynk pada *smartphone*. NodeMCU dapat diakses pada perangkat *smartphone* yang akan digunakan sebagai remote control. Ketika analog diarahkan ke atas, maka robot akan melaju ke depan, berlaku juga untuk arah yang lain. Ketika tombol 0 derajat pada remote di tekan, robot akan mengambil barang yang ada di depannya, sedangkan untuk melepaskannya digunakan tombol 180 derajat pada remote.

Kemudian dari modul NodeMCU akan menerima sinyal berupa karakter huruf yang dikirim oleh *smartphone* menggunakan aplikasinya dan kemudian diteruskan ke L298N. L298N ini akan menerima data dari NodeMCU berupa karakter yang kemudian akan menggerakkan Grabber Robot. Setiap komponen pada rangkaian memiliki input karakter huruf yang berbeda yang dihasilkan oleh button pada aplikasi pada *smartphone*.



Gambar 3.30: Skematik Rangkaian secara detail

### Program untuk joystick

```

BLYNK_WRITE(V6) (1)
{
  int x = param[0].asInt(); (2)
  int y = param[1].asInt();

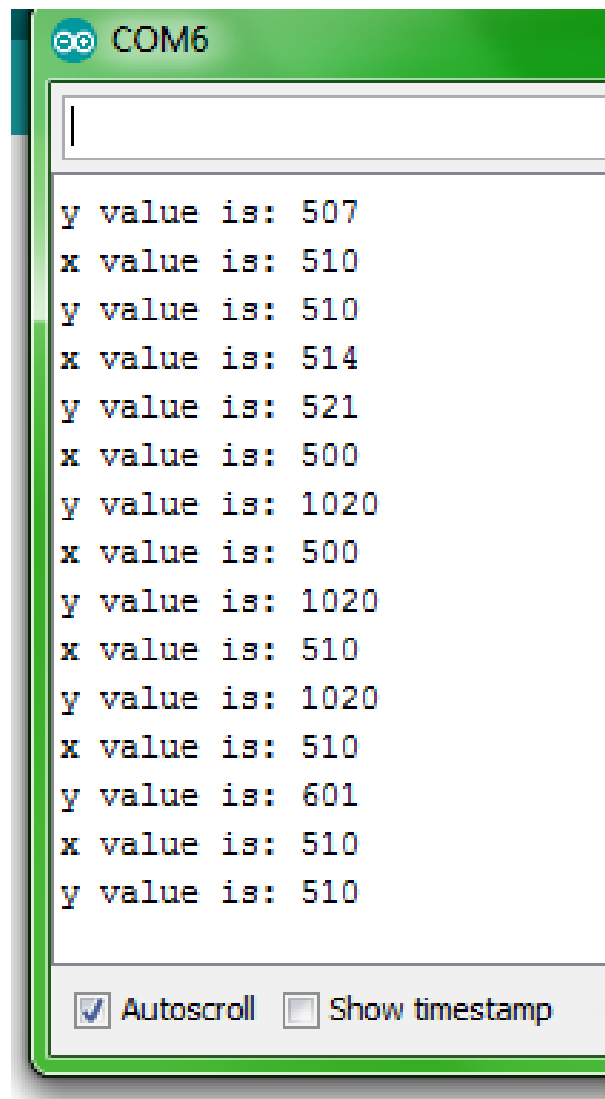
  Serial.print("x value is: "); (3)
  Serial.println(x);
  Serial.print("y value is: ");
  Serial.println(y);

  moveControl(x,y); (4)
}

```

Pada bagian ini merupakan sebuah program untuk mengendalikan joystick pada motor dc. Berikut beberapa bagian mengenai penjelasan tentang program pada sebuah joystick:

1. Fungsinya sebagai membaca data dari virtual pin.
2. Pada bagian ini merupakan sebuah program untuk joystick yang menggunakan "Merge". Jika menggunakan "Merge" maka keluarannya menggunakan 1 pin dan menggunakan "param" pada program arduino IDE.
3. Menampilkan nilai joystick yang berupa x dan y, yang dimana nilainya tersebut dapat di lihat pada serial monitor, seperti pada gambar 3.31.
4. Pada bagian ini, x dan y adalah output yang akan menampilkan monitor arduino IDE.



Gambar 3.31: Nilai x dan y pada sebuah serial monitor di arduino IDE

### Program untuk menjalankan motor dc ketika akan bergerak maju lurus

```

void moveControl(int x, int y) (1)
{
    if(y >= maxRange && x >= minRange && x <= maxRange) (2)
    {
        digitalWrite(IN_1, LOW); (3)
        digitalWrite(IN_2, HIGH); (4)

        digitalWrite(IN_3, HIGH); (5)
        digitalWrite(IN_4, LOW); (6)
    }
}

```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak maju lurus. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. integer bernilai x dan y
2. Output dari program ini akan membuat motor dc bergerak maju lurus.
3. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *LOW*.
4. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *HIGH*.
5. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *HIGH*.
6. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *LOW*.

### Program untuk menjalankan motor dc ketika akan belok kanan

```

else if (x >= maxRange && y >= maxRange) {      (1)

    digitalWrite(IN_1, LOW);                        (2)
    digitalWrite(IN_2, HIGH);                      (3)

    digitalWrite(IN_3, LOW);                        (4)
    digitalWrite(IN_4, HIGH);                      (5)
}

```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak belok kanan. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc belok kanan.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *LOW*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *HIGH*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *LOW*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *HIGH*.

### Program untuk menjalankan motor dc ketika akan belok kiri

```
else if(x <= minRange && y >= maxRange){  
  
    digitalWrite(IN_1, HIGH);  
    digitalWrite(IN_2, LOW);  
  
    digitalWrite(IN_3, HIGH);  
    digitalWrite(IN_4, LOW);  
  
}
```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak belok kiri. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc belok kiri.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *HIGH*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *LOW*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *HIGH*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *LOW*.

### Program untuk menjalankan motor dc ketika akan berhenti

```
else if (y < maxRange && y > minRange &&  
x < maxRange && x > minRange){          (1)  
  
    digitalWrite(IN_1, LOW);              (2)  
    digitalWrite(IN_2, LOW);              (3)  
  
    digitalWrite(IN_3, LOW);              (4)  
    digitalWrite(IN_4, LOW);              (5)  
  
}
```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk berhenti. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc berhenti.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *LOW*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *LOW*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *LOW*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *LOW*.

### Program untuk menjalankan motor dc ketika akan bergerak mundur

```

else if (y <= minRange && x >= minRange && x <= maxRange){ (1)

    digitalWrite(IN_1, HIGH);           (2)
    digitalWrite(IN_2, LOW);           (3)

    digitalWrite(IN_3, LOW);           (4)
    digitalWrite(IN_4, HIGH);          (5)
}

```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak mundur. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc bergerak mundur.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *HIGH*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *LOW*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *LOW*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *HIGH*.



**Program untuk menjalankan motor dc ketika akan bergerak mundur  
dengan belok kanan**

```
else if(y <= minRange && x <= minRange){ (1)

    digitalWrite(IN_1, LOW); (2)
    digitalWrite(IN_2, HIGH); (3)

    digitalWrite(IN_3, LOW); (4)
    digitalWrite(IN_4, HIGH); (5)

}
```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak mundur dengan belok kanan. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc bergerak mundur dengan belok kanan.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *LOW*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *HIGH*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *LOW*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *HIGH*.

**Program untuk menjalankan motor dc ketika akan bergerak mundur  
dengan belok kiri**

```
else if(y <= minRange && x >= maxRange){ (1)

    digitalWrite(IN_1, HIGH); (2)
    digitalWrite(IN_2, LOW); (3)

    digitalWrite(IN_3, HIGH); (4)
    digitalWrite(IN_4, LOW); (5)

}
```

Pada bagian ini merupakan sebuah program untuk menggerakkan motor dc untuk bergerak mundur dengan belok kiri. Berikut penjelasan mengenai beberapa bagian tentang program pada sebuah motor dc :

1. Output dari program ini akan membuat motor dc bergerak mundur dengan belok kiri.
2. Pin digital ini berjenis PWM dan output dari motor dc 1 akan bernilai *HIGH*.
3. Pin digital ini berjenis PWM dan output dari motor dc 2 akan bernilai *LOW*.
4. Pin digital ini berjenis PWM dan output dari motor dc 3 akan bernilai *HIGH*.
5. Pin digital ini berjenis PWM dan output dari motor dc 4 akan bernilai *LOW*.

### Tipe data utama untuk penyimpanan bilangan

```
int minRange = 312;
int maxRange = 712;
```

- int minRange

tipe data ini bersifat integer, dengan memiliki *minimal Range* 312 yang digunakan untuk menjalankan motor dc.

- int maxRange

tipe data ini bersifat integer, dengan memiliki *maximal Range* 712 yang digunakan untuk menjalankan motor dc.

### Library untuk L298N *motor driver*

```
#define IN_1 15          (1)
#define IN_2 13          (2)
#define IN_3 2           (3)
#define IN_4 0           (4)
```

1. Pada l298N ini menggunakan pin IN 1 dan L298N ini terhubung ke NodeMCU dengan menggunakan GPIO 15 (D8)
2. Pada l298N ini menggunakan pin IN 2 dan L298N ini terhubung ke NodeMCU dengan menggunakan GPIO 13 (D7)
3. Pada l298N ini menggunakan pin IN 3 dan L298N ini terhubung ke NodeMCU dengan menggunakan GPIO 2 (D4)
4. Pada l298N ini menggunakan pin IN 4 dan L298N ini terhubung ke NodeMCU dengan menggunakan GPIO 0 (D3)

**Program untuk *horizontal slider* yang mengendalikan *base servo***

```
BLYNK_WRITE(V1)                (1)
{  int a = param.asInt();        (2)
  servo1.write(a);               (3)
}
```

1. Fungsinya sebagai membaca data dari virtual pin.
2. int a yang artinya nilainya bersifat integer, dan "*param*" ini di gunakan untuk membaca fungsi dari virtual pin.
3. servo 1 ini adalah *base servo*

**Program untuk *vertical slider* yang mengendalikan *elbow servo***

```
BLYNK_WRITE(V2)                (1)
{  int a = param.asInt();        (2)
  servo2.write(a);               (3)
}
```

1. Fungsinya sebagai membaca data dari virtual pin.
2. int a yang artinya nilainya bersifat integer, dan "*param*" ini di gunakan untuk membaca fungsi dari virtual pin.
3. servo 2 ini adalah *elbow servo*

**Program untuk *vertical slider* yang mengendalikan *shoulder servo***

```
BLYNK_WRITE(V3)                (1)
{  int a = param.asInt();        (2)
  servo3.write(a);               (3)
}
```

1. Fungsinya sebagai membaca data dari virtual pin.
2. int a yang artinya nilainya bersifat integer, dan "*param*" ini di gunakan untuk membaca fungsi dari virtual pin.

3. servo 3 ini adalah *shoulder servo*

**Program untuk button 0 derajat yang mengendalikan *gripper servo***

```
BLYNK_WRITE(V4){ (1)
  int state = param.asInt(); (2)
  if (state == 1) { (3)
    servo4.write(0); (4)
  }
}
```

1. Fungsinya sebagai membaca data dari virtual pin.
2. int a yang artinya nilainya bersifat integer, dan "*param*" ini di gunakan untuk membaca fungsi dari virtual pin.
3. Statement 1 menandakan robot bergerak.
4. servo 4 ini adalah *gripper servo* dan output dari program ini ialah untuk melepaskan barang yang ada di depannya.

**Program untuk button 180 derajat yang mengendalikan *gripper servo***

```
BLYNK_WRITE(V5){ (1)
  int state = param.asInt(); (2)
  if (state == 1) { (3)
    servo4.write(180); (4)
  }
}
```

1. Fungsinya sebagai membaca data dari virtual pin.
2. int a yang artinya nilainya bersifat integer, dan "*param*" ini di gunakan untuk membaca fungsi dari virtual pin.
3. Statement 1 menandakan robot bergerak.
4. servo 4 ini adalah *gripper servo* dan output dari program ini ialah untuk mengambil barang yang ada di depannya.

### Program untuk mendeklarasikan servo

<code>servo1.attach(5);</code>	(1)
<code>servo2.attach(4);</code>	(2)
<code>servo3.attach(14);</code>	(3)
<code>servo4.attach(12);</code>	(4)
<code>Servo servo1;</code>	(5)
<code>Servo servo2;</code>	(6)
<code>Servo servo3;</code>	(7)
<code>Servo servo4;</code>	(8)

1. Servo1.attach ini yang menentukan variabel servo 1 pada GPIO 5 di NodeMCU.
2. Servo2.attach ini yang menentukan variabel servo 2 pada GPIO 4 di NodeMCU.
3. Servo3.attach ini yang menentukan variabel servo 3 pada GPIO 14 di NodeMCU.
4. Servo4.attach ini yang menentukan variabel servo 4 pada GPIO 12 di NodeMCU.
5. Servo.servo1 ini ialah untuk menentukan servo yang digunakan yaitu *base servo*.
6. Servo.servo1 ini ialah untuk menentukan servo yang digunakan yaitu *elbow servo*.
7. Servo.servo1 ini ialah untuk menentukan servo yang digunakan yaitu *shoulder servo*.
8. Servo.servo1 ini ialah untuk menentukan servo yang digunakan yaitu *gripper servo*.

### Program untuk menjalankan aplikasi Blynk

<code>void setup()</code>	(1)
<code>{</code>	
<code>  Serial.begin(9600);</code>	(2)
<code>  Blynk.begin(auth, ssid, pass);</code>	(3)

#### 1. void setup

Void Setup ini merupakan semua kode yang ada di void setup akan dibaca sekali oleh Arduino atau mikrokontroler lainnya yang dapat menggunakan software arduino.

#### 2. Serial.begin(9600)

untuk menentukan kecepatan pengiriman dan penerimaan data melalui port serial. Kecepatan yang umum digunakan adalah 9600 bit per detik (9600 bps). Namun, kecepatan hingga 115.200 detik didukung oleh Arduino Uno.

#### 3. Blynk.begin(auth, ssid, pass)

untuk mengaktifkan program IoT pada arduino IDE, agar blynk dapat terhubung ke jaringan WiFi.

**Library ini untuk menjalankan *remote control* robot lengan pada aplikasi *blynk***

```
#define BLYNK_PRINT Serial          (1)
#include <ESP8266WiFi.h>              (2)
#include <BlynkSimpleEsp8266.h>      (3)
#include <Servo.h>                    (4)
```

Berikut penjelasan mengenai library yang digunakan untuk menjalankan *remote control* robot lengan pada aplikasi *blynk*:

1. BLYNK \_ PRINT Serial

library ini digunakan untuk membaca dan menerima program dari Arduino IDE.

2. ESP8266WiFi.h

library ini digunakan untuk menghubungkan ESP8266 dengan WiFi agar dapat menjalankan *Grabber Robot*.

3. BlynkSimpleEsp8266.h

library ini digunakan untuk menghubungkan ESP8266 dengan Blynk.

4. servo.h

library ini digunakan untuk servo.



### Sebagai penghubung antara NodeMCU dengan aplikasi blynk

```
char auth [] = "7d786b75eb914657ba029a78faff50d1";  
char ssid [] = "qwerty";  
char pass [] = "qwerty123";
```

Dalam memprogram IoT menggunakan *remote control blynk* diwajibkan menggunakan *auth*, *ssid*, dan *pass* sebagai penghubung antara program arduino IDE dengan aplikasi blynk.

- char auth

Ini merupakan autentikasi yang di dapatkan di bagian *project settings* dan di kirim melalui email. Autentikasi ini berisfat char (*character*), tipe data char adalah tipe data yang mengambil satu byte memori yang menyimpan suatu nilai karakter. Karakter harfiah ditulis dalam kutip tunggal, misalnya 'A' dan untuk multi karakter digunakan tanda kutip ganda, seperti "ABC".

- char ssid

Ini merupakan autentikasi yang di dapatkan di bagian *project settings* dan di kirim melalui email. Autentikasi ini berisfat char (*character*), tipe data char adalah tipe data yang mengambil satu byte memori yang menyimpan suatu nilai karakter. Karakter harfiah ditulis dalam kutip tunggal, misalnya 'A' dan untuk multi karakter digunakan tanda kutip ganda, seperti "ABC". *char ssid* ini nama dari WiFi yang digunakan.

- char pass

Ini merupakan autentikasi yang di dapatkan di bagian *project settings* dan di kirim melalui email. Autentikasi ini berisfat char (*character*), tipe data char adalah tipe data yang mengambil satu byte memori yang menyimpan suatu nilai karakter. Karakter harfiah ditulis dalam kutip tunggal, misalnya 'A' dan untuk multi karakter digunakan tanda kutip ganda, seperti "ABC". *char pass* ini *password* dari WiFi yang digunakan.

### **void loop**

```
void loop ()  
{  
  
  Blynk.run ();  
}
```

- **void loop**  
void loop ini digunakan untuk menjalankan suatu siklus program, yang akan dilakukan terus-menerus hingga mikrokontroler yang digunakan mati/reset.
- **Blynk.run**  
Digunakan untuk mengkomunikasikan dengan blynk server.