LAPORAN PROJECT

**“SIMULATOR VENDING MACHINE”**



**Disusun oleh :**

**Andi Mei Prasetyo Isworo**

**(1110171035)**

**Program Studi Teknik Elektronika**

**Departemen Elektro**

**Politeknik Elektronika Negeri Surabaya**

**Surabaya**

**2019**

**Daftar Isi**

[**I.** **Pendahuluan** 3](#_Toc26159970)

[**II.** **Tujuan** 3](#_Toc26159971)

[**III.** **Dasar Teori** 3](#_Toc26159972)

[**IV.** **Metodologi Percobaan** 3](#_Toc26159973)

[**A.** **Bahan Percobaan** 3](#_Toc26159974)

[**B.** **Desain *Hardware*** 4](#_Toc26159975)

[***C.*** **Desain *Software*** 7](#_Toc26159976)

[**V.** **Analisa Data** 8](#_Toc26159977)

[**A.** **State machine** 8](#_Toc26159978)

[**B.** **Multi state** 8](#_Toc26159979)

[**C.** **Perhitungan Timer 2** 8](#_Toc26159980)

[**D.** **Program** 8](#_Toc26159981)

[**VI.** **Kesimpulan** 10](#_Toc26159982)

[**VII.** **Link Repository GitHub** 10](#_Toc26159983)

# **Pendahuluan**

Pada project ini dibuat suatu alat simulator vending machine dengan harga satu item yaitu permen seharga 1000 rupiah. Terdapat 3 buah actuator yang disimulasikan menggunakan 3 buah LED utama (LED drop, LED 500, LED 1000). Sedangkan untuk input terdapat 2 buah push button (PB Proses, PB Batal), untuk sensor koin digunakan sensor cahaya(photodiode) yang sebelumnya telah dibuat pada project sebelumnya.

# **Tujuan**

Setelah menyelesaikan project ini, yang didapat adalah:  
• Dapat menjelaskan definisi sistem multi-state berdasar waktu  
• Dapat menjelaskan definisi sistem multi-state berdasar input dan waktu  
• Dapat membuat program dengan sistem multi-state

# **Dasar Teori**

Dua katagori pada sistem multi-state secara umum adalah:

* Sistem multi-state berdasar waktu

Ciri dari sistem ini adalah, perpindahan antar state (keadaan) hanya bergantung pada jalannya waktu.Sebagai contoh, sebuah sistem yang mungkin dimulai dari keadaan A, secara berulang menjalankan subroutine1, selama 10 detik. Kemudian pindah ke keadaan B selama 5 detik untuk mengerjakan subroutine2. Selanjutnya dapat pindah ke keadaan A kembali, dan seterusnya. Sistem kontrol lampu lalu lintas adalah contohnya.

* Sistem multi-state berdasar input dan waktu.

Ini adalah bentuk sistem yang lebih dikenal, dimana perpindahan antar keadaan bergantung pada jalannya waktu daninput dari sistem. Sebagai contoh, sebuah sistem mungkin hanya akan pindah dari keadaan A ke keadaan B jika sebuah input tertentu diterima dalam waktu X detik. Mesin cuci tadi adalah contoh dari sistem ini, sepertin halnya sistem alarm anti-pencuri bahkan sampai kepada sistem autopilot pada pesawat terbang.Sebagai pelengkap, ada satu lagi kemungkinan dari sistem multi-state, yaitu:

* Sistem multi-state berdasar input.

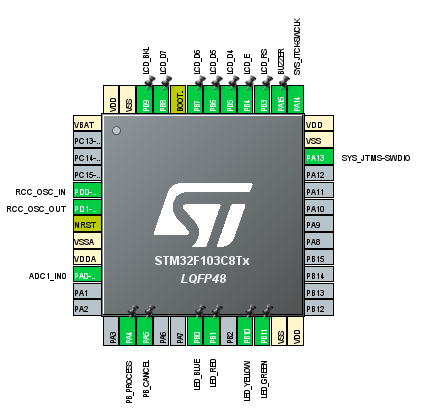
Sistem ini sangat jarang dipakai, dimana perpindahan antar keadaan hanya bergantung pada input sistem. Sebagai contoh, sebuah sistem hanya akan pindah dari keadaan A ke keadaan B jika sebuah input  
tertentu diterima. Jika input tidak diterima maka sistem akan terus berada di keadaan A selamanya. Karena sistem ini tidak memiliki konsep waktu maka tidak ada cara untuk menerapkan timeout atau sejenisnya. Sistem seperti ini tidak dibahas dalam modul.

# **Metodologi Percobaan**

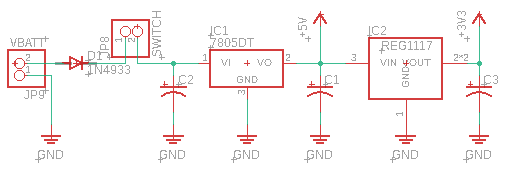
## **Bahan Percobaan**

* + - 1. STM32F103C8 (*Blue Pill*) + ST LINK v2 1 buah
      2. Push Button 2 buah
      3. LCD karakter 16x2 1 buah
      4. Photo diode 3mm 1 buah
      5. Infra Red 3mm 1 buah
      6. PCB Polos 1 buah
      7. Kabel jumper secukupnya
      8. Kabel pita secukupnya
      9. Kertas karton 3mm secukupnya
      10. Header Male & Female secukupnya
      11. Baut + Mur secukupnya

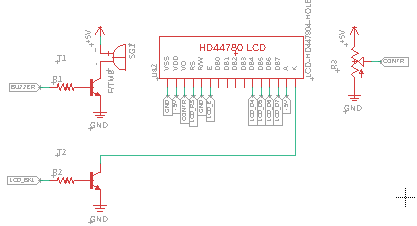
## **Desain *Hardware***



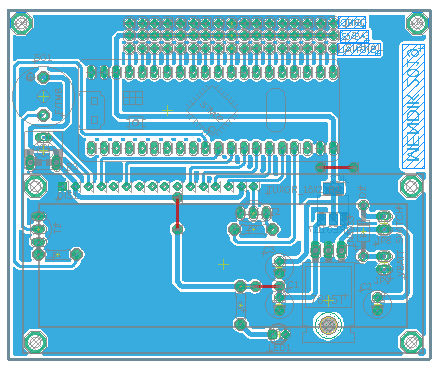
*Gambar 1.1 Konfigurasi Pin yang digunakan*



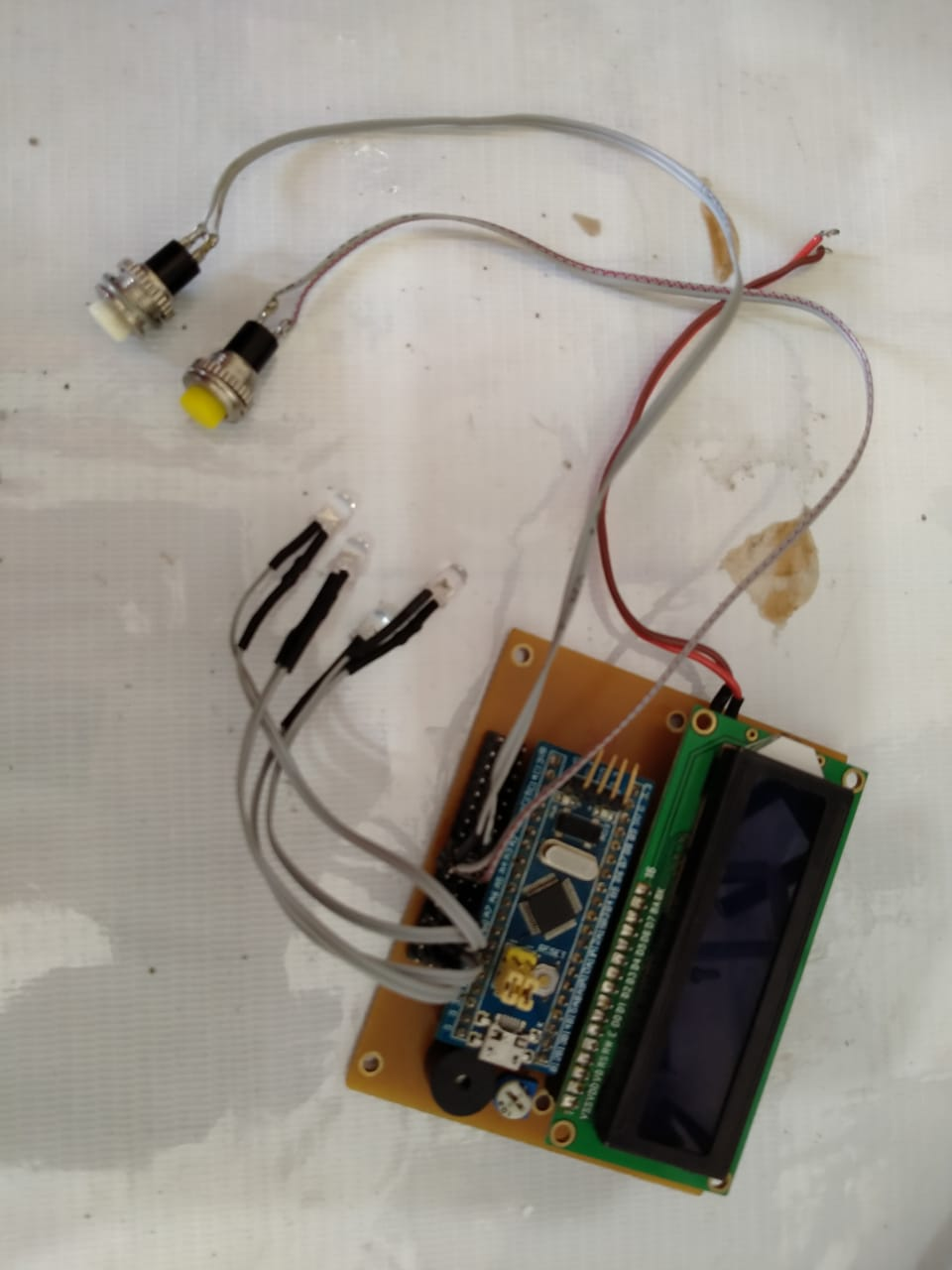
*Gambar 1.2 Rangkaian Regulator Power Supply*



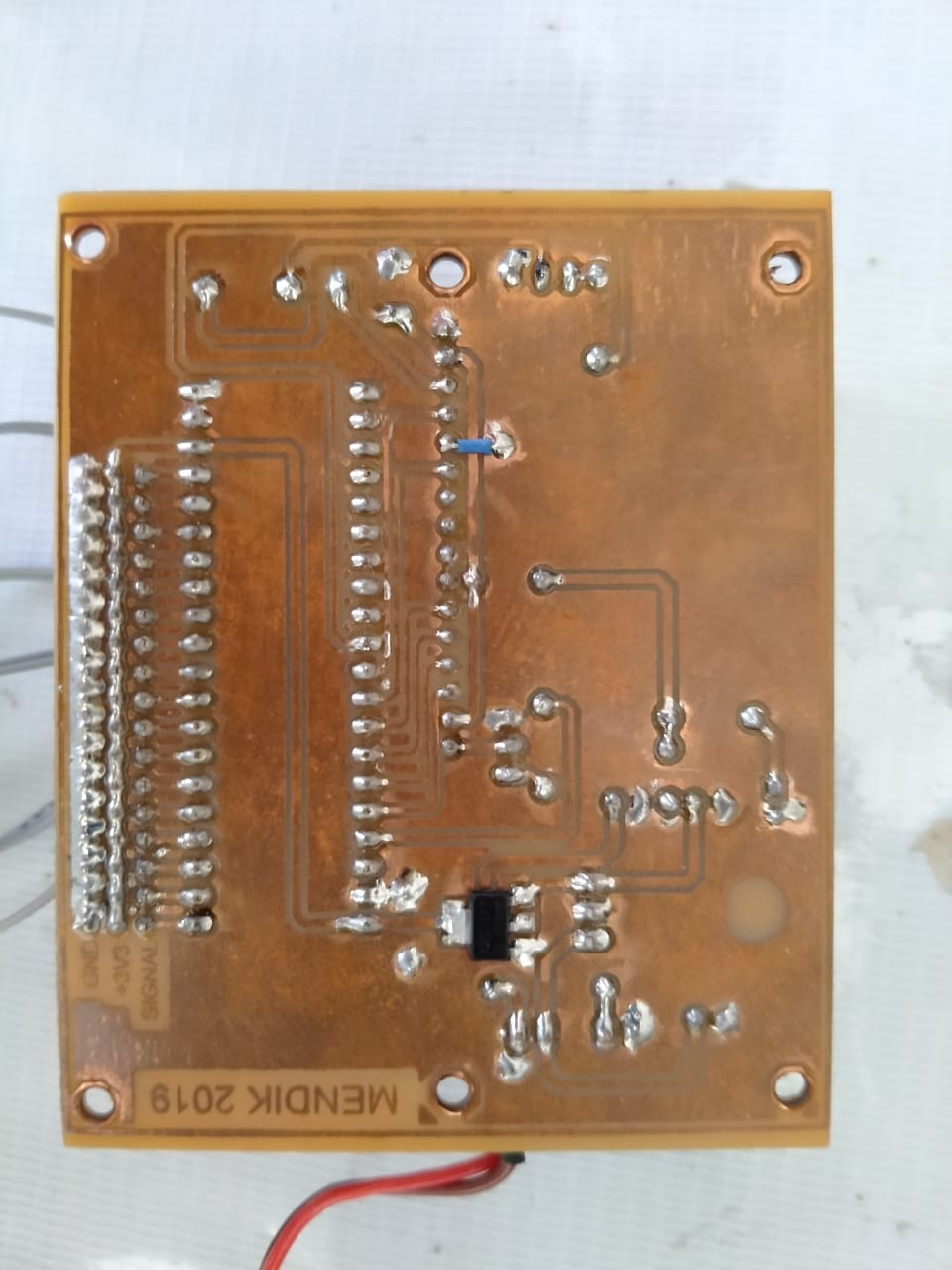
*Gambar 1.3 Rangkaian LCD dan Buzzer*



*Gambar 1.4 Layout Main PCB*



*Gambar 1.5 Rangkaian Simulator Vending Machine*



*Gambar 1.6 PCB Tampak Bawah*



*Gambar 1.7 Alat tampak atas*



*Gambar 1.8 Alat Keseluruhan*

## **Desain *Software***

Pada rancangan yang penulis buat terdapat 8 buah state machine yang akan diimplememtasikan pada hardware.

RESET

Koin 500

Koin 500

Koin 1000

Batal

Batal

Proses

Koin 500

Koin 1000

*Gambar 2.1 State machine*

# **Analisa Data**

## **State machine**

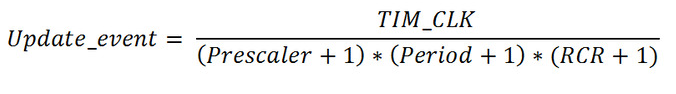
Pada project ini menggunkan state machine moore, dimana kondisi keluaran berada disalam state, sehingga membuat system lebih stabil.

Pada project ini terdapat 8 buah state. Dimana state pertama adalah start yang selanjutnya lompat ke state *insertCoin1,* pada state ini akan lompat ke state *Ready* jika koin yang dimasukkan 1000, jika koin yang dimasukkan 500 maka akan lompat ke state *insertCoin2.* Pada state ini jika dimasukkan koin 500 maka akan lompat ke state K500, yang artinya koin akan dikembalikan 500 dan state akan lompat ke state *Ready.* Jika yang dimasukkan adalah koin 500 maka langsung menuju state *Ready.* Pada state ready dapat diproses dan juga dibatalkan. Jika pada state ini pembeli memasukkan uang maka akan dikembalikan sesuai kelebihan uang yang di mesin.

## **Multi state**

Pada project ini banyak sekali tugas tertetntu yang harus dikerjakan. Jika tanpa adanya algoritma multi state maka akan sulit untuk mengimplementasikan project ini.

## **Perhitungan Timer 2**

Pada timer 2 penulis menginginkan Update\_event dilakukan setiap 500us sekali untuk kerperluan keseuruhan *task* yang akan dikerjakan, TIM\_CLK yang telah disetting adalah 72Mhz, berdasarkan persamaan tersebut maka didapatkan nilai Prescaler = 17999 dan Periode = 1 karena auto reload penulis disable maka nilai RCR diabaikan.

## **Program**

Pada project ini program terlalu banyak sehingga penulis hanya melampirkan beberapa program yang penting, untuk lebih lengkapnya dapat dilihat pada link git hub.

Cuplikan program pada state *insertCoin1.*

**case** *insertCoin1*:{

/\* Printed once in loop \*/

**if**(printEn){

printEn=0;

myLCD\_setCursor(0, 1); myLCD\_print("Money: "); myLCD\_printNum(money);

}

/\* Check type of coin when sensor detected, and give appropriate flag \*/

**if**(Detect\_500()){

buzzer(1);

myLCD\_clear();

myLCD\_setCursor(0, 0); myLCD\_print("Rp.500 inserted !");

myFlag=*in500*;

}

**if**(Detect\_1000()){

buzzer(1);

myLCD\_clear();

myLCD\_setCursor(0, 0); myLCD\_print("Rp.1000 inserted");

myFlag=*in1000*;

}

/\* Check flag and goto next state \*/

**if**(myFlag==*in500*){

**if**(++delay > 2000){

delay=0;

buzzer(0);

money+=500;

myLCD\_clear();

myFlag=*noCoin*;

myLCD\_setCursor(0, 0); myLCD\_print("Insert Coin !");

printEn=1;

myState=*insertCoin2*;

}

}

**if**(myFlag==*in1000*){

**if**(++delay > 2000){

delay=0;

buzzer(0);

money+=1000;

myLCD\_clear();

myFlag=*noCoin*;

printEn=1;

myState=*ready*;

}

}

}**break**;

Cuplikan Program Deteksi Koin

**\_Bool** **Detect\_1000**(**void**){

**\_Bool** flagDetect=0;

**if**((myADC\_read() > 300) && (myADC\_read() < 2700))

debounce3 = debounce3<<1;

**else**

debounce3 = (debounce3<<1)|1;

**if**(debounce3==3)

flagDetect=1;

**return** flagDetect;

}

**\_Bool** **Detect\_500**(**void**){

**\_Bool** flagDetect=0;

**if**(myADC\_read() > 3000)

debounce4 = debounce4<<1;

**else**

debounce4 = (debounce4<<1)|1;

**if**(debounce4==3)

flagDetect=1;

**return** flagDetect;

}

State yang digunakan

**enum** state{*start*, *startDelay*, *insertCoin1*, *insertCoin2*, *ready*, *cancel*, *drop*, *k500*, *k1000*, }myState;

Cuplikan Program inisialisasi timer 2 dengan tick 500us

**void** **MX\_TIM2\_Init**(**void**)

{

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

htim2.Instance = TIM2;

htim2.Init.Prescaler = 17999;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 1;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim2) != *HAL\_OK*)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim2, &sClockSourceConfig) != *HAL\_OK*)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig) != *HAL\_OK*)

{

Error\_Handler();

}

}

# **Kesimpulan**

Berdasarkan hasil percobaan dan analisa yang telah penulis lakukan maka dapat ditarik kesimpulan diantaranya.

* Suatu program yang mempunyai banyak kondisi, akan lebih mudah memprogramnya dengan algoritma *Multistate (mealy maupun moore).*
* Suatu state dapat diatur waktunya melalui interrupt timer.
* Jika program tidak membutuhkan waktu yang cepat, maka interrupt timer dapat disetting lambat, hal tersebut dapat menghemat daya yang dikonsumsi *microcontroller.*
* Agar *task schedule* tidak terganggu, atau bahkan dapat membuat *microcontroller stack* maka hendaknya tidak menggunakan delay yang membuat program berhenti pada baris program tertentu atau yang disebut *delay blocking.*

# **Link Repository GitHub**

[*https://github.com/AndiMei/Embedded\_SimulatorVendingMachine.git*](https://github.com/AndiMei/Embedded_SimulatorVendingMachine.git)