UNIVERSITAS INDONESIA

Cloth Mask Vending Machine with Assembly Language

LAPORAN PROYEK AKHIR

KELOMPOK BeagleBoneBlack
SISTEM BERBASIS KOMPUTER - 02

Haidar Hanif	**(1806148694)**
Ivan Widjanarko	**(1806148706)**
Jonathan Elloy	**(1806148712)**
M. As'ad Muyassir	**(1806199953)**

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

MEI 2020

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas berkat dan karunia-Nya, sehingga kami dapat menyelesaikan Laporan Proyek Akhir yang berjudul "Cloth Mask Vending Machine with Assembly Language" yang ditugaskan oleh guru mata kuliah Sistem Berbasis Komputer - 02, yaitu Ibu Dr. Prima Dewi Purnamasari, S.T., M.Sc. dari Universitas Indonesia yang beralamat di Jl. Margonda Raya, Pondok Cina, Beji, Depok, Jawa Barat, 16424.

Laporan proyek akhir ini berisikan deskripsi sistem embedded yang kelompok kami rancang, penjelasan flowchart proyek, lampiran source code, serta lampiran link video YouTube. Selain itu, akan dijelaskan juga mengenai latar belakang pemilihan proyek, pembagian tugas, definisi singkat perangkat yang digunakan, serta sekamtik rangkaian dari proyek kelompok kami.

Akhirnya, "Tiada gading yang tak retak". Oleh karena itu, saran dan kritik yang membangun sangat kami harapkan demi kesempurnaan pada tugas kami selanjutnya. Semoga laporan proyek akhir ini dapat bermanfaat dan memberikan banyak informasi kepada para pembaca. Terima kasih.

Depok, 14 Mei 2020

Kelompok BeagleBoneBlack

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada tahun 2020, pandemi virus COVID-19 merajalela di seluruh belahan bumi, termasuk di Indonesia. Dilansir dari [1], hingga tanggal 11 Mei 2020 pukul 16.21 WIB (Waktu Indonesia bagian Barat), masyarakat Indonesia yang telah dinyatakan positif mencapai 14.265 penduduk, di mana jumlah masyarakat yang sembuh adalah 2.881 penduduk dan jumlah masyarakat yang meninggal adalah 991 penduduk. Untuk mengatasi hal ini, pemerintah Indonesia berusaha memutus mata rantai penyebaran virus COVID-19 dengan menerapkan beberapa hal, seperti Physical Distancing, Self Isolation, serta PSBB (Pembatasan Sosial Berskala Besar). Selain itu, masyarakat juga dianjurkan untuk menjaga kebersihan diri, rajin mencuci tangan, hindari memakan daging mentah, dan memakai masker kain ketika bepergian keluar rumah.

Mengetahui hal ini, tugas proyek akhir mata kuliah Sistem Berbasis Komputer – 02 untuk program Studi Teknik Komputer, Departemen Teknik Elektro, Fakultas Teknik, Universitas Indoensia diharuskan berkaitan dengan tindakan pencegahan, pemantauan, pengendalian, atau perawatan baik untuk pasien, masyarakat umum bukan pasien, lingkungan, dan sebagainya. Melalui proyek akhir ini, kelompok kami diminta untuk merancang hardware dan software sebuah sistem embedded sederhana dengan memanfaatkan MCU 8051 (atau keluarganya) dan mengimplementasikan program tersebut dengan menggunakan bahasa Assembly [2]. Pada akhirnya, kelompok kami memutuskan untuk membuat suatu program vending machine (mesin penjual otomatis) masker kain dengan bahasa Assembly dan memanfaatkan software Proteus untuk mensimulasikan program tersebut.

1.2 Deskripsi Proyek

Dalam proyek akhir ini, kami berusaha untuk menerapkan beberapa topik kuliah terkait, seperti The 8051 Microcontroller & Hardware Connection, 8051 Assembly Language Programming, Jump, Loop and Call Instructions, I/O Port Programming, 8051 Addressing Modes, Arithmetic & Logic Instructions, Timer Programming, serta Serial Programming [2]. Input dari mesin vending machine ini adalah nomor NPM pembeli masker yang akan dimasukkan oleh pembeli dengan Matrix Keypad. Untuk outputnya, vending machine ini memiliki 3 buah perangkat keras output, yaitu 16X2 LCD, LED Dot Matrix, dan Servo Motor. 16X2 LCD dan LED Dot Matrix akan menampilkan informasi apakah pembeli berhasil

mendapatkan masker kain atau tidak. Sementara itu, Servo Motor akan menggerakan masker kain, sehingga masker tersebut bisa diperoleh dan digunakan oleh pembeli dari vending machine tersebut untuk mencegah penularan virus COVID-19.

Langkah-langkah pembelian masker kain dengan vending machine ini adalah sebagai berikut. Pertama, pembeli akan memasukkan NPM dengan menggunakan Matrix Keypad. Input pembeli tersebut akan diperiksa dan dicocokkan dengan data yang ada di dalam ROM. Jika input yang dimasukkan oleh pembeli sama dengan data yang ada di dalam ROM, maka Servo Motor motor akan digerakkan dan masker akan diterima oleh pembeli. Selain itu, terdapat juga tulisan yang menandakan bahwa masker berhasil diperoleh pada 16X2 LCD serta simbol O pada LED Dot Matrix. Jika input yang dimasukkan oleh pembeli tidak sama dengan data yang ada di dalam ROM, maka akan terdapat tulisan yang menandakan bahwa input pembeli salah pada 16X2 LCD serta simbol X pada LED Dot Matrix. Penjelasan lebih lengkapnya akan dijelaskan pada bagian Skematik Rangkaian.

1.3 Pembagian Tugas

Pembagian tugas dan tanggung jawab dari setiap anggota kelompok kami adalah sebagai berikut.

Nama	NPM	Tugas dan Tanggung Jawab
Haidar Hanif	1806148694	Servo Motor dan Video Simulasi
Ivan Widjanarko	1806148706	LED Dot Matrix dan Laporan Proyek Akhir
Jonathan Elloy	1806148712	16x2 LCD dan Edit Video
M. As'ad Muyassir	1806199953	Matrix Keypad dan Video Penjelasan Code

Tabel 1. Tugas dan Tanggung Jawab

BAB II
LANDASAN TEORI

2.1 Bahasa Assembly

Setiap prosesor yang ada pada komputer atau laptop hanya memahami instruksi bahasa mesin yang terdiri dari kombinasi 1 dan 0 (angka biner). Dengan instruksi bahasa mesin ini yang dimengerti oleh prosesor ini, maka komputer dapat melakukan berbagai proses komputasi, seperti membaca input dari user, memproses input yang dimasukkan (arithmatic, logic, control, dll), serta menampilkan output hasil pemrosesan tersebut. Namun, instruksi mesin ini tidaklah sederhana dan cukup sulit dipahami oleh manusia. Oleh karena itu, dibuatlah bahasa assembly yang mampu menerjemahkan bahasa tingkat rendah (low-leve language) menjadi bahasa mesin (machine language).

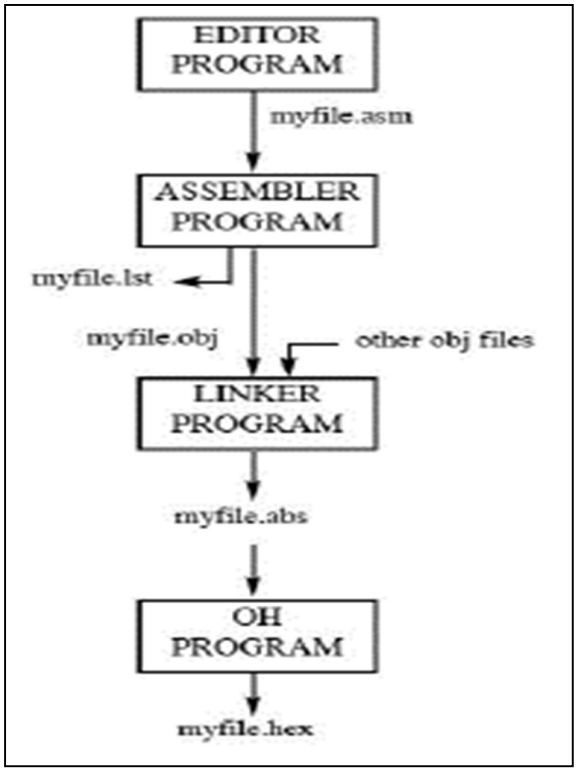
Dengan memahami bahasa assembly, maka kita dapat mengetahui bagaimana suatu instruksi mengakses dan memproses data, bagaimana prosesor mengakses dan mengeksekusi intruksi tersebut, bagaimana data ditampilkan pada memori dan perangkat eksternal lainnya, bagaimana program mengakses perangkat eksternal, serta bagaimana program berinteraksi dengan sistem operasi, prosesor, dan BIOS. Bahasa assembly ini memiliki beberapa keuntungan. Bahasa assembly sangat cocok untuk pekerjaan yang sangat memeperhatikan waktu. Bahasa assembly hanya menggunakan sedikit memori dan waktu eksekusi yang cepat. Selain itu, bahasa assembly juga cocok digunakan untuk interrupt dan program memori lainnya.

Dalam proyek akhir ini, mikrokontroler yang digunakan adalah mikrokontroler 8051 atau keluarganya [3]. Dalam mikrokontroler 8051, terdapat beberapa 8-bit register, seperti register A (Accumulator), B, R0, R1, R2, R3, R4, R5, R6, dan R7. Selain itu, terdapat juga 16-bit register, seperti DPTR dan PC (Program Counter). Tidak hanya itu, terdapat juga beberapa register bank pada mirkoprosesor 8051, yaitu Bank 0 (00H – 07H), Bank 1 (08H – 0FH), Bank 2 (10H – 17H), serta Bank 3 (18H – 1FH) yang masing-masing memiliki 8 register berukuran 8-bit. Intruksi pada bahasa assembly terbagi atas 4 bagian, yaitu label, mnemonic, operands, dan comment.

Untuk membuat suatu program dalam bahasa assembly, maka akan terdapat beberapa langkah-langkah sebagai berikut.

a. Diperlukan sebuah file .asm yang berisikan sorce code bahasa assembly.
b. File .asm tersebut kemudian di-fed ke 8051 assembler yang kemudian akan mengkonversikannya ke kode mesin, sehingga dihasilkan file .obj dan file .lst.

c. Satu atau beberapa file .obj akan diproses oleh 8051 linker yang kemudian akan dihasilkan file .abs.
d. File .abs tersebut kemudian akan di-fed ke program yang disebut dengan OH Converter (Object to Hexadecimal), sehingga dihasilkan file .hex yang siap di-burn pada ROM.



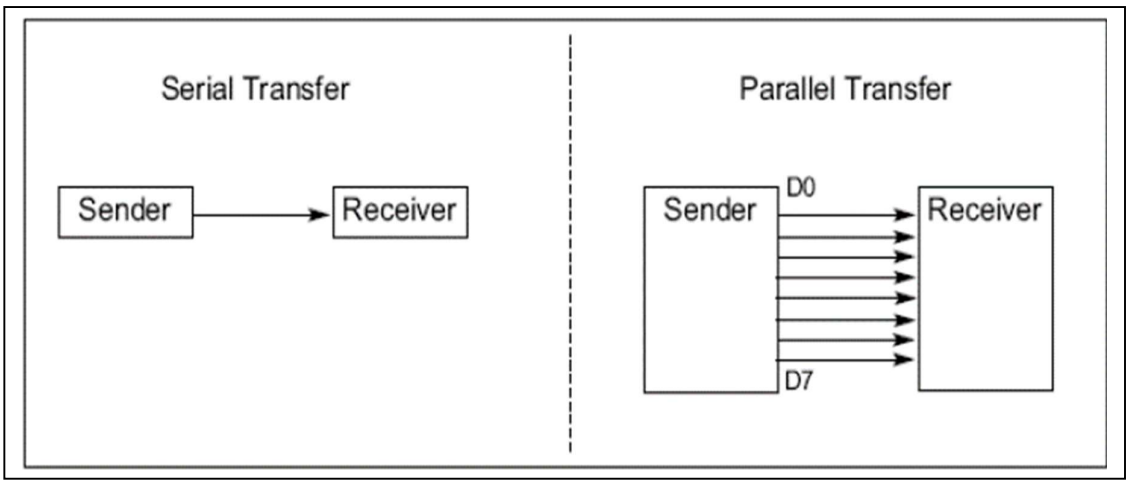
Gambar 1. Langkah Membuat Program

Intruksi yang paling sering digunakan pada bahasa assembly adalah MOV. Intruksi ini berfungsi untuk menyalin suatu nilai dari satu register / memori ke register / memori lainnya. Selain itu, terdapat juga instruksi jump (lompat), seperti JZ, JNZ, DJNZ, CJNE, JC, JNC, JB, JNB, JBC, SJMP, LJMP, ACALL, LCALL, dll. Intruksi jump ini dapat digunakan untuk intruksi looping (pengulangan) maupun memamnggil subroutine. Mikrokontroler 8051 dapat menerapkan beberapa instruksi untuk komputasi arithmetic dan logic. Beberapa instruksi yang sering digunakan adalah ADD, ADDC, DAA, SUBB, MUL, DIV.

Pada mikrokontroler 8051, terdapat 4 buah port yang dapat digunakan agar perangkat keras eksternal dapat terhubung dengan mikrokontroler tersebut, yaitu Port 0, Port 1, Port 2, dan Port 3. Masing-masing port tersebut terdiri atas 8 bit (bit 7 sebagai MSB dan bit 0 sebagai LSB) di mana masing-masing bit tersebut dapat diprogram tersendiri. Dalam penggunaan port ini, sering digunakan intruksi SETB (set ke HIGH), CLR (set ke LOW), serta CPL (complement).

Terdapat 5 buah mode pengalamatan (addressing mode) pada mikrokontroler 8051, yaitu immdiate addressing, register addressing, direct addressing, register indirect addressing, serta indexed addressing. Immediate addressing merupakan salah satu mode pengalamatan di mana suatu nilai secara langsung disalin ke register yang ditandai dengan tanda pound (#). Register addressing merupakan salah satu mode pengalamatan antar regsiter di mana ukuran register asal dan tujuan harus sama, misalnya register tujuan adalah register 8-bit, maka register asal harus berukuran 8-bit juga. Demikian pula untuk register 16-bit. Direct addressing merupakan salah satu mode pengalamatan di mana tujuan dari pengalamatan adalah suatu alamat yang ada pada RAM. Mode pengalamatan ini sering digunakan pada stack ketika menggunakan instruksi PUSH ataupun POP. Register indirect addressing merupakan salah satu mode pengalamatan di mana nilai dalam suatu regsiter menjadi alamat tujuan pengalamatan yang ditandai dengan tanda @. Indexed addressing mode merupakan salah satu mode pengalamatan untuk mengakses data dari Look-Up Tabel (LUT) yang terletak pada ROM. Intruksi yang digunakan untuk mode pengalamatan ini adalah MOVC A, @A+DPTR.

Untuk menerapkan delay pada mikrokontroler 8051, dapat diterapkan 2 cara, yaitu dengan menggunakan software based delay dan dengan menggunakan timer programming. Software based delay pada dasarnya menggunakan looping untuk menciptakan delay. Sementara itu, timer programming menggunakan timer yang ada pada mikrokontroler 8051. Ada 2 jenis timer, yaitu Timer 0 dan Timer 1. Masing-masing timer tersebut memiliki 4 mode, yaitu 13-bit timer mode, 16-bit timer mode, 8-bit auto reload, serta split timer mode. Selain untuk delay, timer programming juga dapat digunakan untuk counter.



Gambar 2. Komunikasi Serial vs Paralel

Selain fungsi-fungsi yang sudah dijelaskan di atas, mikrokontroler 8051 juga dapat melakukan komunikasi atau pengiriman data secara serial. Komunikasi serial lebih murah daripada komunikasi paralel karena menggunakan kabel / jalur yang lebih sedikit. Namun, komunikasi serial lebih lambat daripada komunikasi paralel. Untuk komunikasi serial, terdapat 2 metode, yaitu asynchronous dan synchronous. Metode asynchronous mengirimkan data satu per satu dalam suatu waktu, sedangkan metode synchronous mengirimkan sejumlah data dalam suatu waktu.

2.2 Sistem Embedded

Sistem embedded merupakan suatu sistem tertanam yang terdiri atas berbagai komponen yang saling berkaitan satu sama lain. Sistem embedded dapat berupa suatu sistem tersendiri atau bagian dari sistem yang besar. Sistem embedded dapat dianggap sebagai suatu perangkat keras komputer dengan perangkat lunak yang tertanam di dalamnya. Komponen utama dari sistem embedded adalah hardware, software, serta RTOS (Real Timing Operating System). Sifat dari sistem embedded adalah microprocessor and microcontroller based, digerakkan oleh software, dapat diandalkan (reliable), dan sistem pengendali real-time. Sistem embedded memiliki beberapa karakteristik antara lain sebagai berikut. [4]

a. Single-functioned

   Artinya, sistem embedded hanya melakukan fungsi khusus / tertentu

b. Tightly Constrained

   Artinya, batasan metrik (biaya, ukuran, daya, kinerja) pada sistem embedded akan sangat ketat / terbatas

c. Reactive and Real Time

   Artinya, sangat dinamis mengikuti perubahan lingkungan luar dan memproses data secara real time (tanpa delay)

d. Microprocessor or Microcontroller Based

   Artinya, sistem embedded berbasis μp atau μc dan dikendalikan oleh main processing core, seperti DSP (Digital Signal Processor)

e. Memory

   Artinya, sistem embedded sangat memerlukan memori utama, khususnya ROM; namun tidak memerlukan memori sekunder

f. Connected

   Artinya, sistem embedded memiliki komponen tambahan yang menghubungkan sistem dengan perangkat I/O

g. HW-SW System

   Artinya, dalam sistem embedded, hardware digunakan untuk performa dan keamanan sedangkan software digunakan untuk fitur dan fleksibilitas.

h. Bahasa Hardware

   Artinya, Bahasa yang digunakan merupakan bahasa hardware, seperti Ada, VHDL, Verilog, System C, Java, dll

Keuntungan dari sistem embedded adalah mudah untuk dikustomisasi, cepat dan harganya murah, penggunaan daya rendah, serta peforma yang dapat ditingkatkan. Sementara iru, sistem embedded juga memiliki beberapa kerugian, seperti sulit dalam pengembangan dan jarang ditemui di pasaran. Sistem embedded ini banyak diterapkan / diaplikasikan, seperti pada ATM, handphone, kamera digital, dll. Ada beberapa jenis sistem embedded sebagai berikut.

a. Stand Alone

   Dapat bekerja sendiri (otomatis) dalam menerima input, memproses, dan menghasilkan output. Contohnya adalah konsol game, MP3 Player, kamera digital, HP, dll.

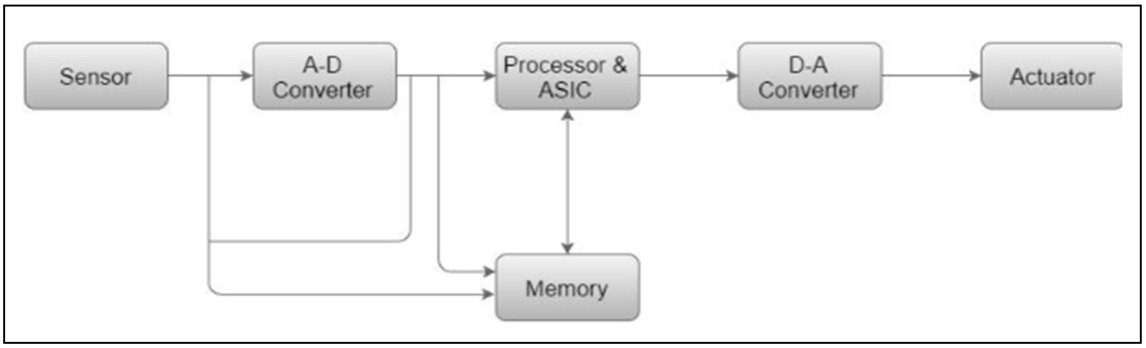
b. Real-Time

   Sangat memperhatikan waktu respon dan memproses pada waktu tertentu. Dibagi menjadi 2 jenis, yaitu Hard Real-Time (keterlambatan waktu tidak dapat ditoleransi. seperti kontrol rudal, kontrol air bag mobil, dll) dan Soft Real-Time (keterlambatan waktu dapat ditoleransi, seperti microwave, mesin cuci, rice cooker, dll)

c. Networked

   Menghubungkan interface jaringan ke sumber akses. Contohnya adalah sistem keamanan rumah melalui LAN yang dihubungkan oleh sistem embedded.

Struktur dari sistem embedded adalah :



Gambar 3. Struktur Sistem Embedded

a. Sensor

   Merupakan alat yang digunakan untuk mengukur kuantitas fisik dan mengubahnya menjadi sinyal listrik sebagai input.

b. A-D Converter

   Merupakan alat yang digunakan untuk mengkonversi sinyal analog menjadi sinyal digital.

c. Processor & ASICs

   Merupakan alat yang digunakan dalam memproses data, yaitu menghasilkan output dan menyimpannya pada memori

d. D-A Converter

   Merupakan alat yang digunakan untuk mengkonversi sinyal digital menjadi sinyal analog

e. Actuator

   Merupakan alat yang digunakan untuk membandingkan antara output dari D-A Converter dengan output yang diharapkan

2.3 Mikrokontroler 8051

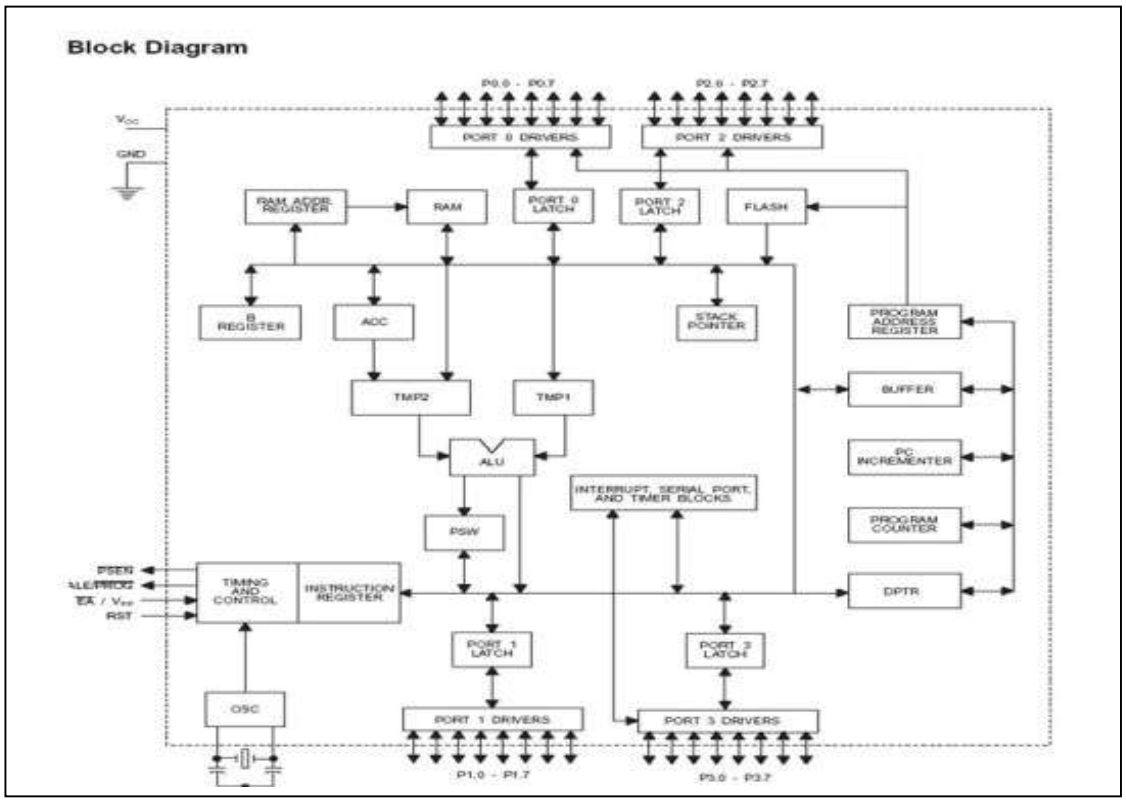
John H. Wharton merupakan seseorang yang berhasil menemukan instruksi dan arsitektur untuk mikrokontroler 8051. Kemudian, pada tahun 1981 Intel Corporation memperkenalkan mikrokontoler ini. Terdapat vendor lain yang juga mengadopsi mikrokontroler Intel 8051, seperti Philips, Siemens, hingga Atmel ATMEL juga membuat mikrokontroler 8051, yaitu mikrokontroler Atmel seri AT89xxx, misalnya AT89S51 dan AT89S52. [5]

Mikrokontroler ini memiliki ROM berukuran 4K byte, RAM berukuran 128 byte, 1 port serial, 2 timer, serta 4 buah port I/O yang masing-masing berukuran 8-bit. Dengan demikian, mikrokontroler ini memiliki CPU yang bekerja pada data berukuran 8-bit pada satu waktu. Mikrokontroler ini termasuk dalam keluarga intel MCS-51 yang mempunyai berbagai macam perangkat dan versi. Umumnya setiap anggota keluarga MCS-51 juga disebut sebagai mikrokontroler 8051.

Mikrokontroler 8051 memiliki beberapa fitur penting antara lain adalah sebagai berikut.

a. MCS-51 merupakan mikrokontoler Intel yang didesain dengan menggunakan teknologi HMOS.
b. Mikrokontroler 8051 bekerja pada frekuensi operasi 12 Mhz serta tersedia dalam versi ROM / EPROM / EEPROM.

c. Mikrokontroler 8051 memiliki 64K memori data dan 64K memori program yang terpisah.
d. Tersedia intruksi perkalian / multiply (MUL) dan pembagian / divider (DIV).
e. Mendukung operasi bitwise dan memiliki prosesor boolean.
f. Tersedia juga dalam versi CHMOS
g. 32 port I/O yang bisa digunakan baik sebagai empat port 8-bit atau 32 I/O
h. Memiliki 16-bit bus alamat yang termutipleksi dengan Port 1 dan Port 2. Port 0 juga sebagai bus data



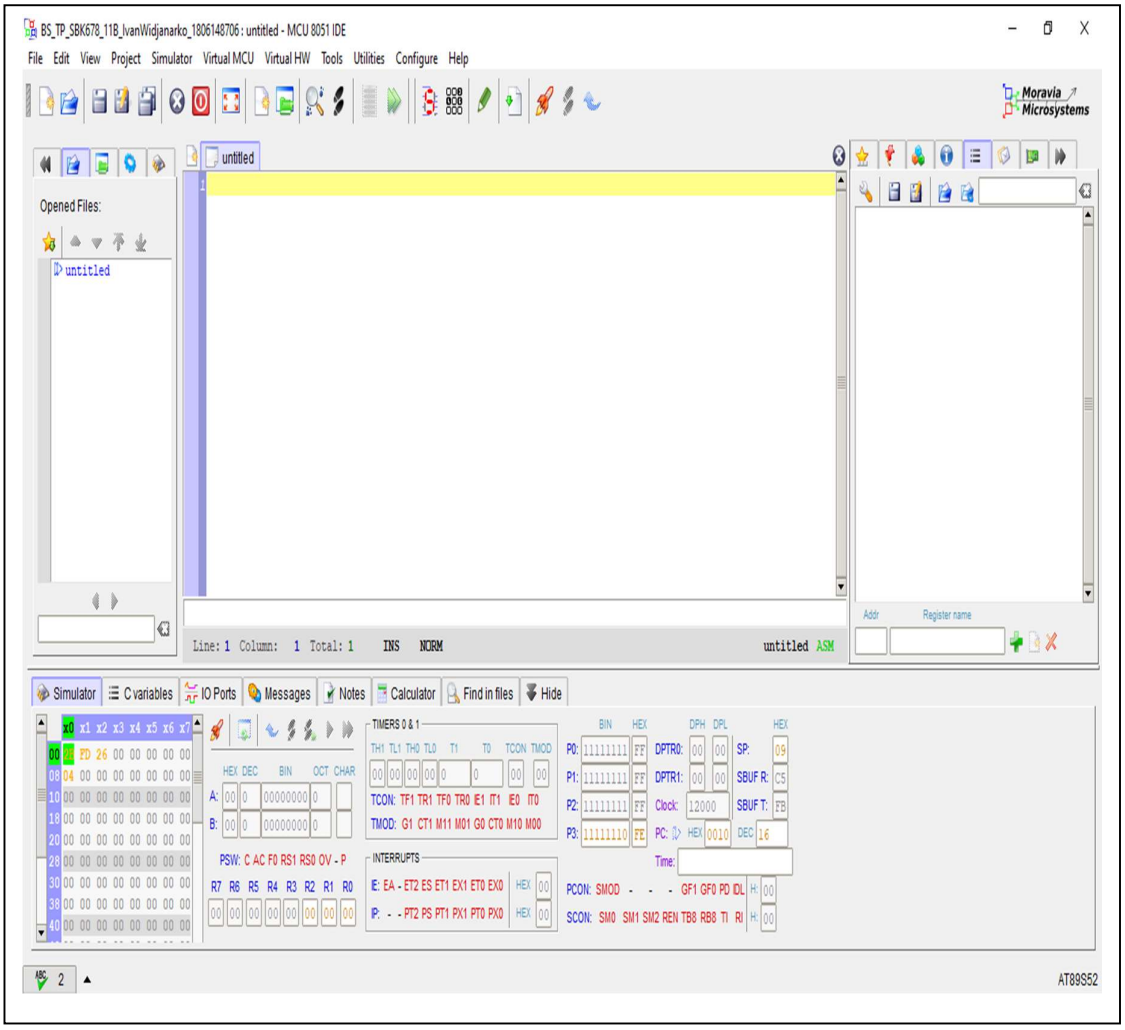
Gambar 4. Blok Diagram Mikrokontroler 8051

Mikrokontroler model lama berarsitektur 8051 masih dipelajari karena susunan arsitektrunya masih sederhana, sehingga dapat dipelajari dengan mudah. Untuk memahami sesuatu yang rumit, maka diperlukan pemahaman mengenai dasar-dasar yang sederhana terlebih dahulu. Dengan demikian, jika kita mempelajari mikrokontroler 8051 yang arsitekturnya masih sederhana, maka diharapkan ke depannya mampu memahami juga mikrokontroler yang lebih rumit aristekturnya.

Dari segi fungsi, pada mikrokontroler 8051, operasi logika boolean tingkatan-bit dapat dilakukan secara langsung dan secara efisien dalam RAM dan register internal, sehingga mikrokontroler 8051 digunakan dalam rancangan awal PLC (Programmable Logic Control).

Selain itu, mikrokontroler 8051 memiliki empat set register yang terpisah, sehingga dapat mempercepat latency interrupt.

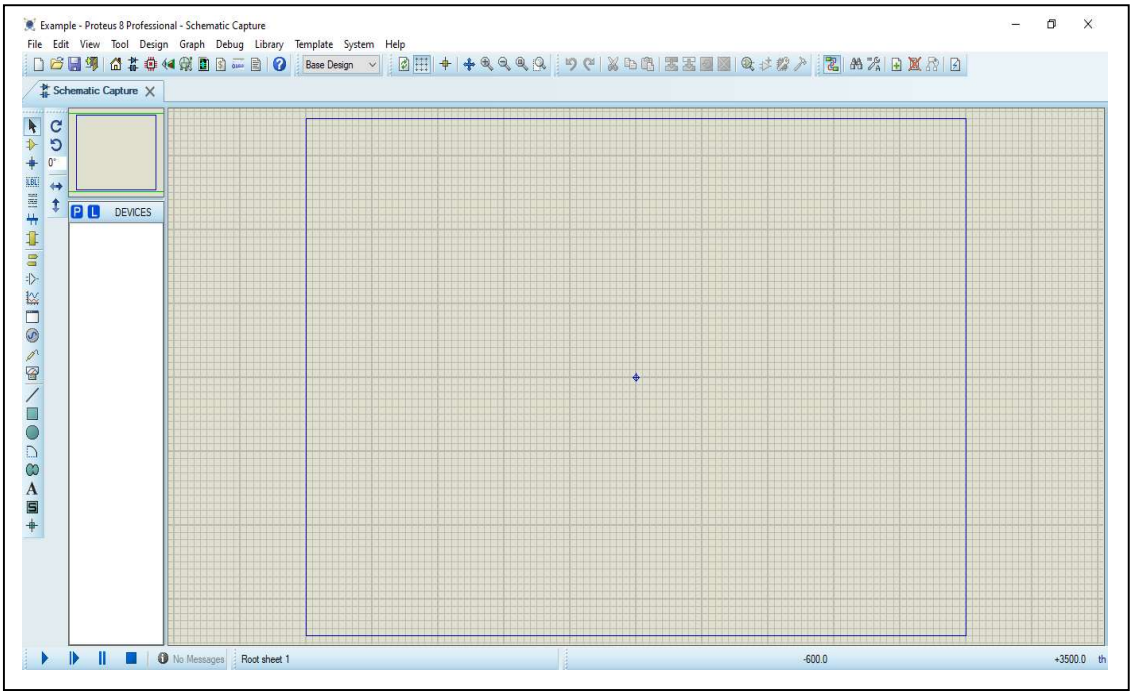
2.4 MCU 8051 IDE



Gambar 5. MCU 8051 IDE

MCU 8051 IDE (Integrated Development Environment) merupakan sebuah perangkat lunak gratis berupa environment untuk mikrokontroler 8051. MCU 8051 IDE memiliki simulator dan assmebler tersendiri di dalamnya. MCU 8051 ini mendukung pemrograman dengan 2 bahasa, yaitu bahasa assembly maupun bahasa C.

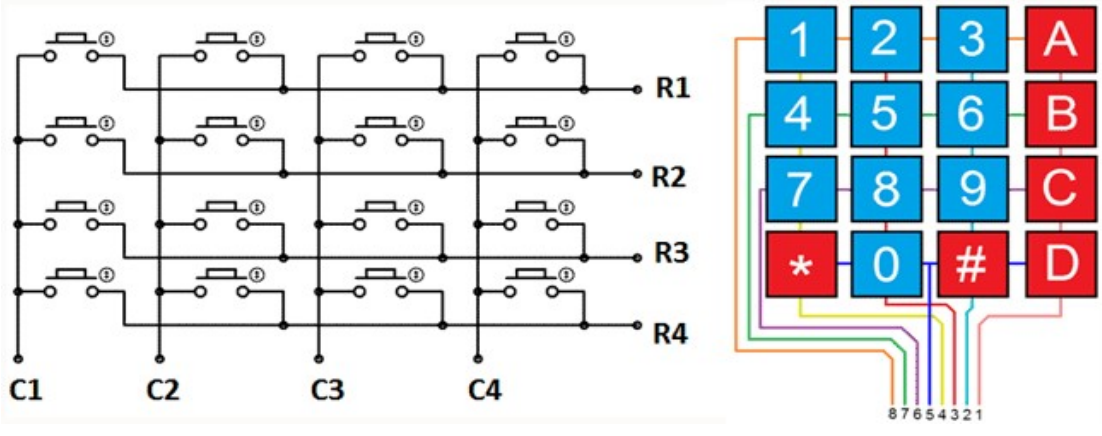
2.5 Proteus



Gambar 6. Proteus

Proteus merupakan salah satu aplikasi / perangkat lunak yang dapat digunakan, baik untuk merangkai ataupun mensimulasikan rangkaian sistem digital. Tidak hanya itu, Proteus juga dapat digunakan untuk membuat PCB layout dan IOT Builder [6]. Proteus memiliki beragam library yang berisikan banyak komponen-komponen digital. Pada proyek kami ini, rangkaian akan dirangkai dan disimulasikan pada Proteus, bukan MCU 8051 IDE. Hal ini dikarenakan tidak terdapat perangkat keras Servo Motor pada MCU 8051 IDE.

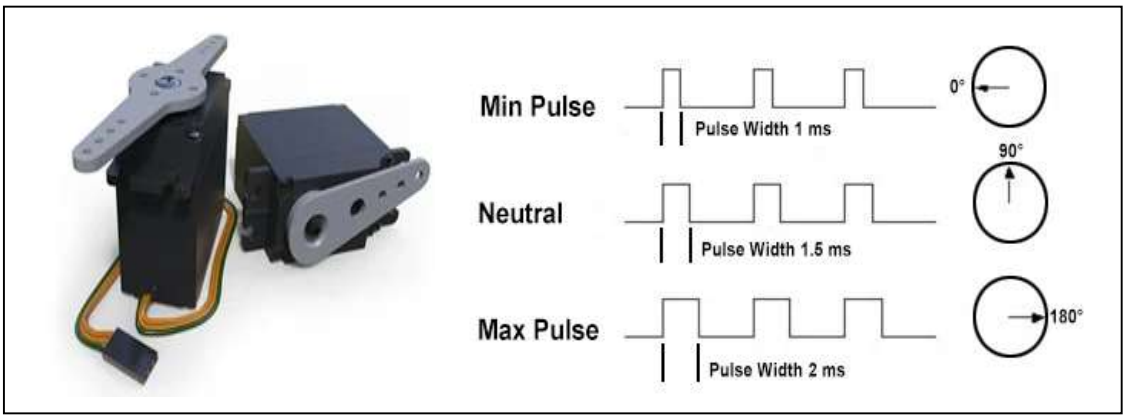
2.6 Matrix Keypad



Gambar 7. Matrix Keypad

Matrix Keypad merupakan sebuah perangkat keras input yang dapat dihubungkan ke mikrokontroler 8051 yang terdiri atas 16 tombol. Ke-16 tombol tersebut adalah angka 0-9, huruf A-D, serta tanda * dan #. Semua tombol tersebut disusun sedemikian rupa ke dalam bentuk matrix 4x4. Terdapat 8 kabel / jalur yang terhubung ke Matrix Keypad, yaitu 4 kabel untuk baris (R1-R4) dan 4 kabel untuk kolom (C1-C4). Program akan mengidentifikasikan tombol mana yang ditekan dengan metode column scanning, yaitu baris akan selalu diberi nilai 0 (ada juga yang berlaku sebaliknya, yaitu selalu diberi nilai 1) dan kolom akan diperiksa apakah sama-sama low atau sama-sama high. Untuk kasus di mana baris selalu diberi nilai 0, maka kolom akan dicek apakah bernilai 0 juga. Jika tombol ditekan, maka kolom akan bernilai 0. Hal inilah yang menyebabkan program mengetahui bahwa sebuah tombol ditekan pada Matrix Keypad. [7]

2.7 Servo Motor



Gambar 8. Servo Motor

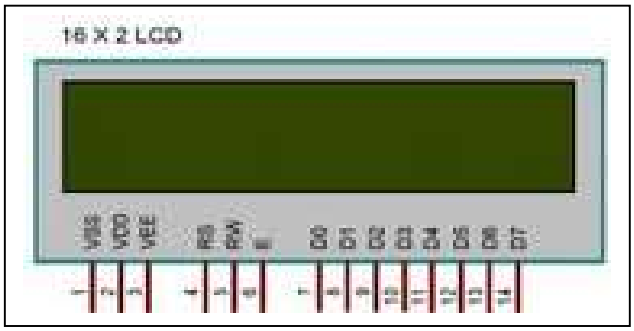
Servo Motor merupakan sebuah perangkat keras output yang dapat dihubungkan ke mikrokontroler 8051 yang dapat menggerakan suatu motor di dalamnya. Motor tersebut dikendalikan dengan sinyal listrik yang menetukan jumlah gerakan poros. Ketika motor berputar, resistansi dari potensiometer akan berubah, sehingga sirkuit kontrol dapat dengan tepat mengatur seberapa banyak gerakan yang ada dan ke arah mana. Ketika poros motor berada pada posisi yang diinginkan, daya yang disuplai ke motor dihentikan. Jika tidak, motor diputar ke arah yang sesuai. Posisi yang diinginkan dikirim melalui pulsa listrik melalui kabel sinyal. Kecepatan motor sebanding dengan perbedaan antara posisi aktual dan posisi yang diinginkan. Jadi jika motor berada di dekat posisi yang diinginkan, ia akan berputar perlahan, jika tidak maka akan berputar cepat. Ini disebut kontrol proporsional. Ini berarti motor hanya akan berjalan sekeras yang diperlukan untuk menyelesaikan tugas yang dihadapi. [8]

2.8 16X2 LCD

16x2 LCD (Liquid Crystal Display) merupakan sebuah perangkat keras output yang dapat dihubungkan ke mikrokontroler 8051 untuk menampilkan karakter. Setiap karakter berukuran 5x7 pixel matrix dan ada juga yang berukuran 5x8 pixel matrix. 16x2 LCD memiliki 16 buah pin, yaitu VSS, VCC, VEE, RS, R/W, E, DB0-DB7, LED+, serta LED-. Untuk command, terdapat 13 command umum yang dapat digunakan untuk menggunakan 16x2 LCD ini. Ke-13 command tersebut adalah sebagai berikut. [9]

Command	Fungsi
0F	LCD dan kursor menyala
01	Meng-clear-kan layar
02	Kembali ke sisi paling kiri
04	Menggeser kursor ke kiri
06	Menggeser kursor ke kanan
05	Tampilan digeser kek kanan
07	Tampilan digeser ke kiri
0E	Layar menyala dan kursor kedap-kedip
80	Kursor dipindahkan ke paling awal dari baris pertama
C0	Kursor dipindahkan ke paling awal baris kedua
38	Menggunakan kedua baris dan 5x7 pixel matrix
83	Kursor di baris pertama posisi ketiga
3C	Menghidupkan baris kedua
08	Layar dan kursor mati
0C	Layar menyala dan kursor mati
0C3	Jump ke baris kedua posisi ketiga
0C1	Jump ke baris pertama posisi pertama

Tabel 2. Command untuk 16x2 LCD



Gambar 9. 16x2 LCD

Terdapat 3 langkah utama untuk menggunakan 16x2 LCD pada mikrokontroler 8051, yaitu antara lain sebagai berikut.

a. Inisialisasi LCD

Langkah ini dilakukan dengan menerapkan beberapa command sebagai berikut.

A. 0x38 untuk inisialisasi data 8-bit.

B. 0x0C agar layar LCD menyala dan kursor mati.

C. 0x01 untuk meng-clear-kan tampilan LCD.

D. 0x80 agar kursor berada pada baris pertama.

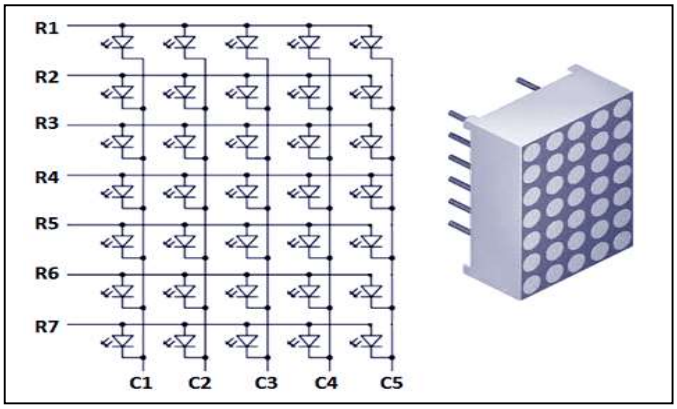
b. Mengirim command ke LCD

Langkah ini dilakukan dengan memberikan nilai 1 / 0 pada pin yang ada di 16x2 LCD. Pin E bernilai 1. Pin RS bernilai 0. Meletakkan data pada regsiter. Pin R/W bernilai 0.

c. Menuliskan data pada LCD

Langkah ini dilakukan dengan memberikan nilai 1 / 0 pada pin yang ada di 16x2 LCD. Pin E bernilai 1. Pin RS bernilai 1. Meletakkan data pada register. Pin R/W bernilai 0.

2.9 LED Dot Matrix

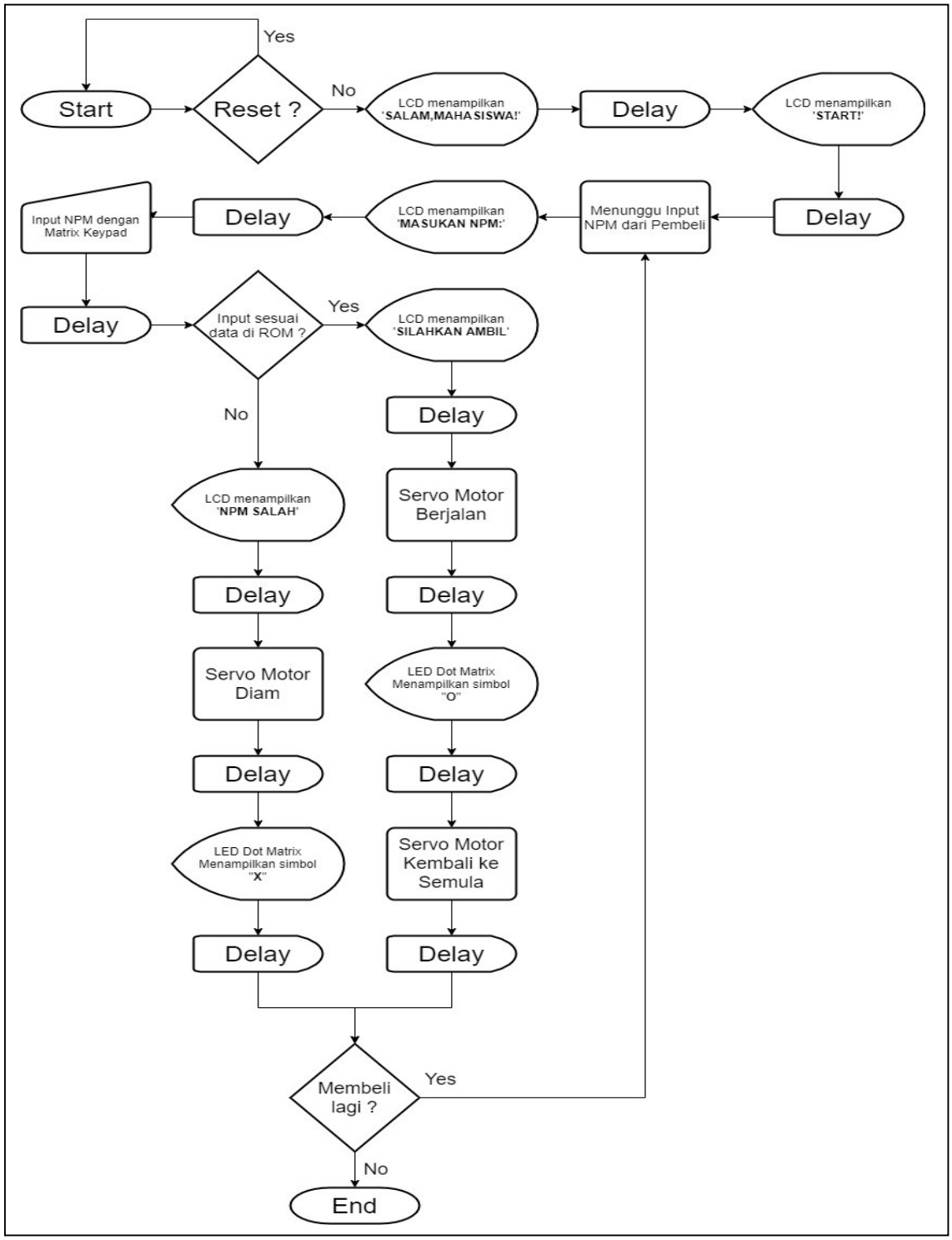


Gambar 10. LED Dot Matrix

LED Dot Matrix merupakan sebuah perangkat keras output yang dapat dihubungkan ke mikrokontroler 8051 di mana terdiri dari beberapa LED yang disusun sedemikian rupa ke dalam suatu kotak. Beberapa jenis LED Dot Matrix yang umum dijumpai adalah 7x5, 7x15, 8x8, dll. LED Dot Matrix biasanya digunakan sebagai penampil output yang tidak terlalu mementingkan resolusi tampilan output tersebut. Cara kerja LED Dot Matrix sama dengan cara kerja Matrix Keypad, yaitu dengan metode column scanning. [10]

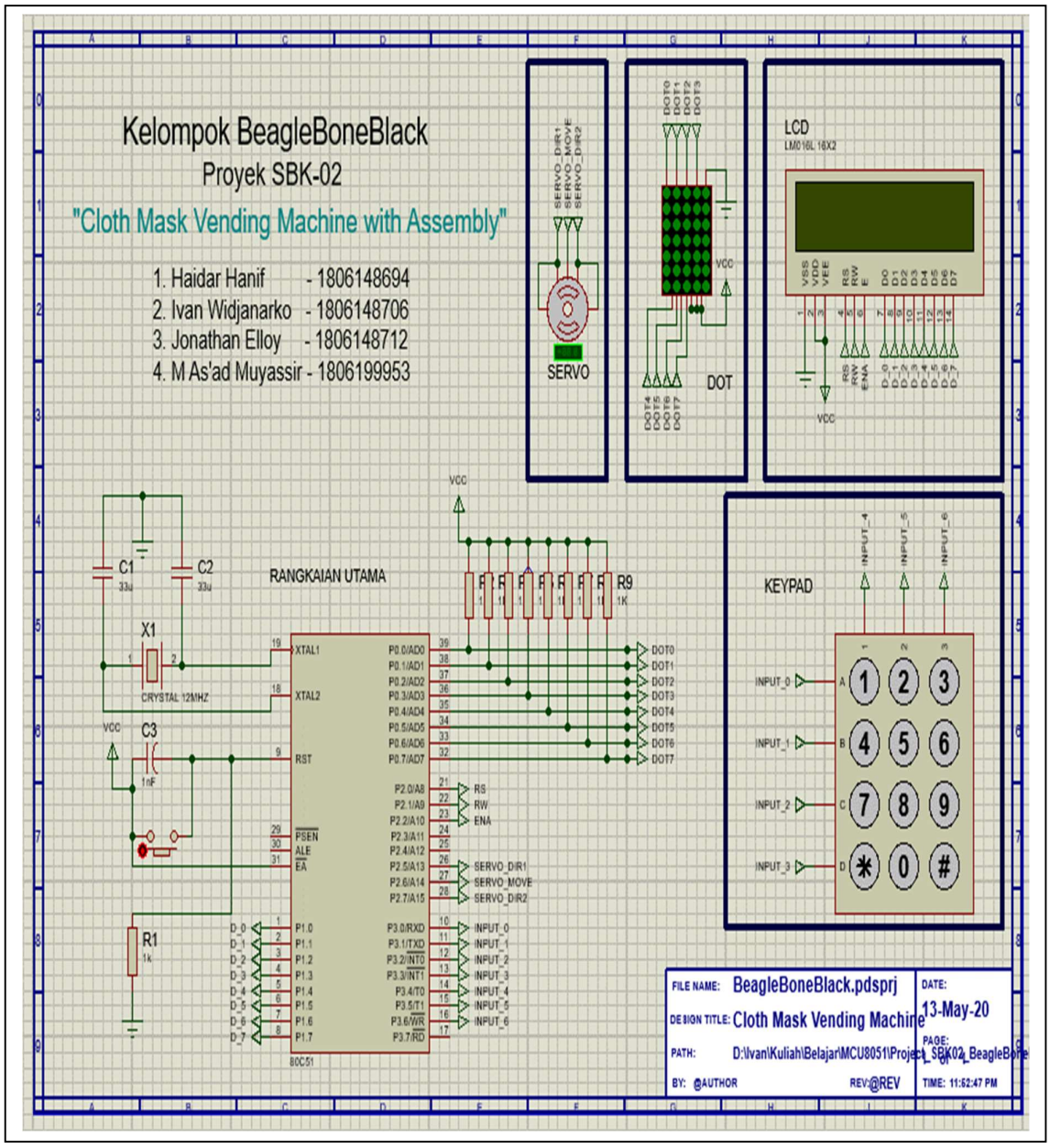
BAB III
PEMBAHASAN

3.1 Flowchart



Gambar 11. Flowchart

3.2 Skematik Rangkaian



Gambar 12. Skematik Rangkaian

Cara kerja Cloth Mask Vending Machine tersebut adalah sebagai berikut. Pertama, LCD akan menampilkan pesan 'SALAM,MAHASISWA!' di baris pertama dan ‘START!’ di baris kedua. Kemudian, tampilan LCD akan di-clear-kan dan menampilkan 'MASUKAN NPM:' pada baris pertama dan siap menampilkan input pembeli di baris kedua. Pembeli akan memasukkan NPM dengan menggunakan Matrix Keypad. Jika tidak ada input, maka program akan terus menunggu hingga terdapat input.

Input dari pembeli tersebut akan ditampilkan pada LCD, lalu diperiksa dan dicocokkan dengan data yang ada di dalam ROM. Jika input yang dimasukkan oleh pembeli sama dengan data yang ada di dalam ROM, maka Servo Motor motor akan digerakkan dan masker akan diterima oleh pembeli. Selain itu, terdapat juga pesan ‘SILAHKAN AMBIL' pada LCD serta simbol O pada LED Dot Matrix. Servo Motor akan dikembalikan ke posisi semula, dan siap menggerakkan masker lagi nantinya. Jika input yang dimasukkan oleh pembeli tidak sama dengan data yang ada di dalam ROM, maka akan terdapat pesan ‘NPM SALAH' pada LCD serta simbol X pada LED Dot Matrix. Pada kondisi ini, Servo Motor tidak akan bergerak sama sekali. Setelah proses ini, maka pembeli dapat membeli lagi masker dengan memasukkan NPM. Program akan terus berjalan dan akan direset jika tombol / pin RST ditekan.

3.3 Penjelasan Source Code

```
INPUT EQU P3                    ; Matrix Keypad menggunakan Port 3 (bit 0 hingga bit 6)
COUNTER EQU 0                   ; Counter di R0
D EQU P1                        ; Pin D pada LCD 16x2 menggunakan Port 1 (bit 0 hingga bit 7)
RS EQU P2.0                     ; Pin RS pada LCD 16x2 menggunakan Port 2 bit 0
RW EQU P2.1                     ; Pin RW pada LCD 16x2 menggunakan Port 2 bit 1
ENA EQU P2.2                    ; Pin E pada LCD 16x2 menggunakan Port 2 bit 2
SERVO_DIR_1 EQU P2.5            ; Pin Direction 1 pada Servo menggunakan Port 2 bit 5
SERVO_MOVE EQU P2.6             ; Pin Move pada Servo menggunakan Port 2 bit 6
SERVO_DIR_2 EQU P2.7            ; Pin Direction 2 pada Servo menggunakan Port 2 bit 7
Port_Matrix EQU P0              ; LED_Matrix menggunakan Port 3 (bit 0-3 untuk Row, bit 4-7 untuk
                                Column)
Clear_1 EQU 0FFH                ; Menset nilai ke FFH
Mode EQU 01H                    ; Mode Timer (TMOD)
Time_A EQU 0FCH                 ; Konstanta untuk delay dengan timer
Time_B EQU 066H                 ; Konstanta untuk delay dengan timer
Clear_0 EQU 00H                 ; Menset nilai ke 00H

   ORG 00H                      ; Instruksi dimulai dari alamat 00H
INIT:
   MOV
   Port_Matrix,#Clear_Delay     ; Mengclearkan LED Matrix
   MOV A, #038H                 ; Menggunakan kedua baris dan setiap karakter berukuran 5x7 pixel pada
                                LCD 16x2
   LCALL perintah_LCD           ; Memanggil subroutine perintah_LCD
   MOV A, #0EH                  ; Layar menyala dan kursor kedap-kedip pada LCD 16x2
   LCALL perintah_LCD           ; Memanggil subroutine perintah_LCD
   MOV A, #01H                  ; Meng-clear-kan layar pada LCD 16x2
   LCALL perintah_LCD           ; Memanggil subroutine perintah_LCD
   MOV A, #080H                 ; Kursor dipindahkan ke paling awal dari baris pertamapada LCD 16x2
   LCALL perintah_LCD           ; Memanggil subroutine perintah_LCD
   MOV DPTR, #TAMPILAN1         ; Menyalin isi dari TAMPILAN1 ke DPTR
   LCALL HEM                    ; Memanggil fungsi HEM
```

```
    MOV A, #0C0H              ; Kursor dipindahkan ke paling awal baris kedua pada LCD 16x2
    LCALL perintah_LCD        ; Memanggil subroutine perintah_LCD
    MOV DPTR, #TAMPILAN2      ; Menyalin isi dari TAMPILAN2 ke DPTR
    LCALL HEM                 ; Memanggil subroutine HEM
    LCALL LOOPDE              ; Memanggil subroutine LOOPDE
INIT2: MOV A, #001H           ; Meng-clear-kan layar pada LCD 16x2
    LCALL perintah_LCD        ; Memanggil subroutine perintah_LCD
    MOV A, #80H               ; Kursor dipindahkan ke paling awal dari baris pertamapada LCD 16x2
    LCALL perintah_LCD        ; Memanggil subroutine perintah_LCD
    MOV DPTR, #TAMPILAN3      ; Menyalin isi dari TAMPILAN3 ke DPTR
    LCALL HEM                 ; Memanggil subroutine HEM
    MOV A, #0C0H              ; Kursor dipindahkan ke paling awal baris kedua pada LCD 16x2
    LCALL perintah_LCD        ; Memanggil fungsi perintah_LCD
    SJMP MAINAN               ; Lompat ke label MAINAN
HEM:                          ; Subroutine HEM
    LCALL wait                ; Memanggil subroutine wait
    CLR A                     ; Mengclearkan Accumulator
    MOVC A, @A+DPTR           ; Menyalin isi dari alamat A+DPTR ke Accumulator
    JZ DONEE                  ; Lompat ke label DONEE jika A = 0
    LCALL print_LCD           ; Memanggil subroutine print_LCD
    INC DPTR                  ; Menambahkan nilai DPTR dengan 1
    SJMP HEM                  ; Lompat ke label HEM
DONEE: RET                    ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
wait:                         ; Subroutine wait
    MOV R1, #CLEAR_1          ; Menyalin 0FFH ke R1
      wow1: MOV R2, #00FH     ; Menyalin 00FH ke R2
      wow2: DJNZ R2, wow2     ; Mengurangi nilai R2 dan lompat ke label wow2 jika R2 = 0
        DJNZ R1, wow1         ; Mengurangi nilai R1 dan lompat ke label wow2 jika R1 = 0
    RET                       ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
perintah_LCD:                 ; Subroutine perintah_LCD
    LCALL wait                ; Memanggil subroutine wait
    MOV D, A                  ; Menyalin isi dari Accumulator ke D
    CLR RS                    ; Menset nilai 0 untuk pin RS pada LCD 16x2
    CLR RW                    ; Menset nilai 0 untuk pin RW pada LCD 16x2
    SETB ENA                  ; Menset nilai 1 untuk pin E pada LCD 16x2
    CLR ENA                   ; Menset nilai 0 untuk pin E pada LCD 16x2
    RET                       ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
print_LCD:                    ; Subroutine print_LCD
    LCALL wait                ; Memanggil subroutine wait
    MOV D, A                  ; Menyalin isi dari Accumulator ke D
    SETB RS                   ; Menset nilai 1 untuk pin RS pada LCD 16x2
    CLR RW                    ; Menset nilai 0 untuk pin RW pada LCD 16x2
    SETB ENA                  ; Menset nilai 1 untuk pin E pada LCD 16x2
    CLR ENA                   ; Menset nilai 0 untuk pin E pada LCD 16x2
    RET                       ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

MAINAN: MOV COUNTER,
#10H                          ; Menyalin 10H ke COUNTER (Alamat 0)
MAIN: LCALL KEYPAD            ; Memanggil subroutine KEYPAD
```

```
    CJNE A, #11, ANGKA      ; Membandingkan nilai Accumulator dengan 11. Jika tidak sama, maka lompat ke label ANGKA
    LJMP MAIN               ; Lompat ke label MAIN

;UNTUK MENYIMPAN NPM
ANGKA: MOV R6, #CLEAR_1     ; Menyalin 0FFH ke R6 untuk delay
    LCALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV R6, #02FH           ; Menyalin 02FH ke R6 untuk delay
    LCALL LOOPSV            ; Memanggil subroutine LOOPSV
    XRL A, #30H             ; Melakukan instruksi XOR agar menjadi ASCII dan kemudian disimpan ke Accumulator
    LCALL print_LCD         ; Memanggil subroutine print_LCD
    MOV @R0, A              ; Menyalin nilai Accumulator ke alamat dari isi RO
    INC COUNTER             ; Menambahkan nilai COUNTER dengan 1
    CJNE R0, #1AH, MAIN     ; Membandingkan nilai R0 dengan 1AH. Jika tidak sama, maka lompat ke label MAIN
    LCALL KOMPER            ; Memanggil subroutine KOMPER
    LJMP INIT2              ; Lompat ke label INIT2

;FUNGSI UNTUK MELAKUKAN COMPARE DENGAN NPM DATA
KOMPER:                     ; Subroutine KOMPER
    MOV DPTR, #NPM          ; Menyalin isi dari NPM ke DPTR
    LUPLUP: MOV COUNTER, #10H   ; Menyalin 10H ke COUNTER
    LUP: CLR A              ; Menset nilai 0 untuk Accumulator
    MOVC A, @A+DPTR         ; Menyalin isi dari alamat A+DPTR ke Accumulator
    MOV 1, @R0              ; Menyalin isi dalam alamat pada R0 ke alamat 1
    CJNE A, 1, KELUP        ; Membandingkan nilai Accumulator dengan nilai di alamat 1. Jika tidak sama, maka lompat ke label KELUP
    INC COUNTER             ; Menambahkan nilai COUNTER dengan 1
    INC DPTR                ; Menambahkan nilai DPTR dengan 1
    CJNE R0, #1AH, LUP      ; Membandingkan nilai R0 dengan 1AH. Jika tidak sama, maka lompat ke label LUP
    MOV A, #01H             ; Meng-clear-kan layar pada LCD 16x2
    LCALL perintah_LCD      ; Memanggil subroutine perintah_LCD
    MOV A, #80H             ; Kursor dipindahkan ke paling awal dari baris pertama pada LCD 16x2
    LCALL perintah_LCD      ; Memanggil subroutine perintah_LCD
    MOV DPTR,#TAMPILAN4     ; Menyalin isi dari TAMPILAN4 ke DPTR
    LCALL HEM               ; Memanggil subroutine HEM
    LCALL OVRES             ; Memanggil subroutine OVRES
    RET                     ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
KELUP: JZ KELK              ; Lompat ke label KELS jika A = 0
    CLR C                   ; Menset nilai 0 untuk C
    CLR AC                  ; Menset nilai 0 untuk AC
    MOV A, #1AH             ; Menyalin 1AH ke Accumulator
    SUBB A, COUNTER         ; Mengurangi A dengan COUNTER dan hasilnya diletakkan di Accumulator
    ADD A, DPL              ; Menambah Accumulator dengan DPL dan hasilnya diletakkan di Accumulator
    MOV DPL, A              ; Menyalin nilai di Accumulator ke DPL
```

```
    SJMP LUPLUP                ; Lompat ke label LUPLUP
KELK: MOV A, #01H              ; Menyalin 01H ke Accumulator
    LCALL perintah_LCD         ; Memanggil subroutine Perintah_LCD
    MOV A, #80H                ; Kursor dipindahkan ke paling awal dari baris pertama pada LCD 16x2
    LCALL perintah_LCD         ; Memanggil subroutine perintah_LCD
    MOV DPTR,#TAMPILAN5        ; Menyalin isi dari TAMPILAN5 ke DPTR
    LCALL HEM                  ; Memanggil subroutine HEM
    MOV R5,#75D                ; Menyalin 75D ke R5 untuk looping
X: LCALL MATRIKX               ; Memanggil subroutine MATRIKX
    DJNZ R5, X                 ; Mengurangi nilai R5 dan lompat ke label X jika R5 = 0
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

;FUNGSI UNTUK MEMBACA KEYPAD
KEYPAD: MOV
INPUT,#CLEAR_1                 ; Menyalin 0FFH ke INPUT
    CLR INPUT.0                ; Menset nilai 0 untuk port INPUT bit 0
    JNB INPUT.4,SATU           ; Jika pada port INPUT bit 4 sama dengan 0, maka lompat ke label SATU
    JNB INPUT.5,DUA            ; Jika pada port INPUT bit 5 sama dengan 0, maka lompat ke label DUA
    JNB INPUT.6,TIGA           ; Jika pada port INPUT bit 6 sama dengan 0, maka lompat ke label TIGA
    JNB INPUT.7,AA             ; Jika pada port INPUT bit 7 sama dengan 0, maka lompat ke label AA
    SETB INPUT.0               ; Menset nilai 1 untuk port INPUT bit 0
    CLR INPUT.1                ; Menset nilai 0 untuk port INPUT bit 1
    JNB INPUT.4,EMPAT          ; Jika pada port INPUT bit 4 sama dengan 0, maka lompat ke label EMPAT
    JNB INPUT.5,LIMA           ; Jika pada port INPUT bit 5 sama dengan 0, maka lompat ke label LIMA
    JNB INPUT.6,ENAM           ; Jika pada port INPUT bit 6 sama dengan 0, maka lompat ke label ENAM
    JNB INPUT.7,BB             ; Jika pada port INPUT bit 7 sama dengan 0, maka lompat ke label BB
    SETB INPUT.1               ; Menset nilai 1 untuk port INPUT bit 1
    CLR INPUT.2                ; Menset nilai 0 untuk port INPUT bit 2
    JNB INPUT.4,TUJUH          ; Jika pada port INPUT bit 4 sama dengan 0, maka lompat ke labe TUJUH
    JNB INPUT.5,LAPAN          ; Jika pada port INPUT bit 5 sama dengan 0, maka lompat ke labelLAPAN
    JNB INPUT.6,SEMBIL         ; Jika pada port INPUT bit 6 sama dengan 0, maka lompat ke label
                               SEMBIL
    JNB INPUT.7,CC             ; Jika pada port INPUT bit 7 sama dengan 0, maka lompat ke label CC
    SETB INPUT.2               ; Menset nilai 1 untuk port INPUT bit 2
    CLR INPUT.3                ; Menset nilai 0 untuk port INPUT bit 3
    JNB INPUT.4,TITIK          ; Jika pada port INPUT bit 4 sama dengan 0, maka lompat ke label TITIK
    JNB INPUT.5,NOL            ; Jika pada port INPUT bit 5 sama dengan 0, maka lompat ke label NOL
    JNB INPUT.6,PAGER          ; Jika pada port INPUT bit 6 sama dengan 0, maka lompat ke labe pager
    JNB INPUT.7,DD             ; Jika pada port INPUT bit 7 sama dengan 0, maka lompat ke label DD
    SETB INPUT.3               ; Menset nili 1 untuk port INPUT bit 3
    MOV A, #11                 ; Menyalin nilai 11 ke Accumulator
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
NOL: MOV A, #0                 ; Menyalin nilai 0 ke Accumulator
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
SATU: MOV A, #1                ; Menyalin nilai 1 ke Accumulator
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
DUA: MOV A, #2                 ; Menyalin nilai 2 ke Accumulator
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
TIGA: MOV A, #3                ; Menyalin nilai 3 ke Accumulator
    RET                        ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
```

```
EMPAT: MOV A, #4          ; Menyalin nilai 4 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
LIMA: MOV A, #5           ; Menyalin nilai 5 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
ENAM: MOV A, #6           ; Menyalin nilai 6 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
TUJUH: MOV A, #7          ; Menyalin nilai 7 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
LAPAN: MOV A, #8          ; Menyalin nilai 8 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
SEMBIL: MOV A, #9         ; Menyalin nilai 9 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
AA: MOV A, #11            ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
BB: MOV A, #11            ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
CC: MOV A, #11            ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
DD: MOV A, #11            ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
TITIK: MOV A, #11         ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
PAGER: MOV A, #11         ; Menyalin nilai 11 ke Accumulator
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali
OVRES:                    ; Subroutine OVRES
    CLR SERVO_DIR_2       ; Menset nilai 0 untuk Port 2 bit 7 agar Servo bergerak
    CLR SERVO_MOVE        ; Menset nilai 0 untuk Port 2 bit 6 agar Servo berputar ke kanan
    ACALL LOOPDE          ; Memanggil subroutine LOOPDE
    ACALL LOOPDE          ; Memanggil subroutine LOOPDE
    MOV R6,#59D           ; Menyalin nilai 59D ke R6 untuk delay
    ACALL LOOPSV          ; Memanggil subroutine LOOPSV
    SETB SERVO_MOVE       ; Menset nilai 1 untuk Port 2 bit 6
    SETB SERVO_DIR_2      ; Menset nilai 1 untuk Port 2 bit 7
    MOV R5,#50D           ; Menyalin nilai 50D ke R5 untuk looping
O: ACALL MATRIKO          ; Memanggil subroutine MATRIKO
    DJNZ R5, O            ; Mengurangi nilai R5 dan lompat ke label O jika R5 = 0

    CLR SERVO_DIR_1       ; Menset nilai 0 untuk Port 2 bit 5 agar Servo kembali ke posisi semula
    CLR SERVO_MOVE        ; Menset nilai 0 untuk Port 2 bit 6 agar Servo berputar ke kiri
    ACALL LOOPDE          ; Memanggil subroutine LOOPDE
    ACALL LOOPDE          ; Memanggil subroutine LOOPDE
    MOV R6,#123D          ; Menyalin nilai 123D ke R6 untuk delay
    ACALL LOOPSV          ; Memanggil subroutine LOOPSV
    SETB SERVO_MOVE       ; Menset nilai 1 untuk Port 2 bit 6
    SETB SERVO_DIR_1      ; Menset nilai 1 untuk Port 2 bit 5
    ACALL LOOPDE          ; Memanggil subroutine LOOPDE
    RET                   ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

MATRIKX:                  ; Subroutine MATRIKX
    MOV Port_Matrix,#0E9H ; Menyalin 0E9H ke Port 0 untuk mencetak simbol X
```

```
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#0D6H   ; Menyalin 0D6H ke Port 0 untuk mencetak simbol X
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#0B6H   ; Menyalin 0B6H ke Port 0 untuk mencetak simbol X
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#079H   ; Menyalin 079H ke Port 0 untuk mencetak simbol X
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#CLEAR_0 ; Menyalin 00H ke Port 0 untuk meng-clear-kan LED Dot Matrix
    RET                     ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

MATRIKO: MOV
Port_Matrix,#0E6H           ; Menyalin 0E6H ke Port 0 untuk mencetak simbol O
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#0D9H   ; Menyalin 0D9H ke Port 0 untuk mencetak simbol O
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#0B9H   ; Menyalin 0B9H ke Port 0 untuk mencetak simbol O
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#076H   ; Menyalin 076H ke Port 0 untuk mencetak simbol O
    MOV R6,#10H             ; Menyalin nilai 10H ke R6 untuk delay
    ACALL LOOPSV            ; Memanggil subroutine LOOPSV
    MOV Port_Matrix,#CLEAR_0 ; Menyalin 00H ke Port 0 untuk men-clear-kan LED Dot Matrix
    RET                     ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

LOOPDE:                     ; Subroutine LOOPDE
    MOV R6,#10D             ; Menyalin nilai 10D ke R6
    ITER: MOV R7,#100D      ; Menyalin nilai 100D ke R7
    LOOP: ACALL DELAY       ; Memanggil subroutine DELAY
    DJNZ R7,LOOP            ; Mengurangi nilai R7 dan lompat ke label LOOP jika R7 = 0
    DJNZ R6,ITER            ; Mengurangi nilai R6 dan lompat ke label ITER jika R6 = 0
    RET                     ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

LOOPSV:                     ; Subroutine LOOPSV
    ACALL DELAY             ; Memanggil subroutine DELAY
    DJNZ R6,LOOPSV          ; Mengurangi nilai R7 dan lompat ke label LOOP jika R7 = 0
    RET                     ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

DELAY: MOV TMOD,#MODE       ; Memilih Mode 1, yaitu timer 16-bit
    MOV TH0,#TIME_A         ; Menyalin nilai 0FCH ke TH0
    MOV TL0,#TIME_B         ; Menyalin nilai 066H ke TL0
    SETB TR0                ; Menset nilai 1 untuk TR0 (menyalakan timer)
HERE: JNB TF0,HERE          ; Jika nilai TF0 tidak sama dengan 0, maka lompat ke label HERE
                            (menghitung mundur timer)
    CLR TR0                 ; Menset nilai 0 untuk TR0 (mematikan timer)
```

```
    CLR TF0                         ; Menset nilai 0 untuk TF0
    RET                             ; Keluar dari subroutine dan nilai PC sebelumnya di pop kembali

;DEFINE VARIABEL
    ORG 0A00H
NPM: DB '1806148694'                ; Data NPM Pertama
    DB '1806148706'                 ; Data NPM Kedua
    DB '1806148712'                 ; Data NPM Ketiga
    DB '1806199953'                 ; Data NPM Keempat
    DB 0   ; Data 0

TAMPILAN1: DB
'SALAM,MAHASISWA!',0                ; Isi dari TAMPILAN1
TAMPILAN2:DB '     START!
',0                                 ; Isi dari TAMPILAN2
TAMPILAN3:DB 'MASUKAN
NPM:',0                             ; Isi dari TAMPILAN3
TAMPILAN4:DB 'SILAHKAN
AMBIL',0                            ; Isi dari TAMPILAN4
TAMPILAN5:DB 'NPM SALAH',0          ; Isi dari TAMPILAN5

END                                 ; Akhir dari program secara keseluruhan
```

3.4 Link YouTube dan GitHub

Link YouTube :

https://bit.ly/Proyek_SBK02_BeagleBoneBlack_Video atau

https://youtu.be/N-QcJWBDcuQ

Link GitHub :

https://bit.ly/Proyek_SBK02_BeagleBoneBlack_Code atau

https://github.com/IvanWidjanarko/Cloth-Mask-Vending-Machine-Assembly-

REFERENSI

[1] G. T. P. P. COVID-19, "Beranda | Gugus Tugas Percepatan Penanganan COVID-19," 2020. [Online]. Available: https://covid19.go.id. [Accessed 11 Mei 2020].

[2] D. P. D. Purnamasari, S.T., M.Sc., "Course: Sistem Berbasik Komputer (Semua Kelas)," 2020. [Online]. Available: https://emas.ui.ac.id/pluginfile.php/570525/mod_resource/content/0/TUGAS%20PROYEK%20MIKON_2020.pdf. [Accessed 11 Mei 2020].

[3] M. A. Mazidi, J. G. Mazidi and R. D. McKinlay, The 8051 Microcontoller and Embedded Systems: Using Assembly and C, 2nd ed., Delhi: Pearson/Prentice Hall, 2006.

[4] S. K. BeagleBoneBlack, "Microcontroller 8051 - BeagleBoneBlack (SBK-02)," 02 April 2020. [Online]. Available: https://www.youtube.com/watch?v=M09ByeXav98. [Accessed 11 Mei 2020].

[5] S. Suhaeb, S.T., M.Pd., Y. A. Djawad, S.T., M.Sc., Ph.D., D. H. Jaya, S.Pd., M.T., R. S.T., M.T., D. S. M.Pd. and A. Risal, A.Md., Mikrokontroler dan Interface, Makassar: Universitas Negeri Makassar, 2017.

[6] L. Electronics, "PCB Design and Circuit Simulator Software for Modern EDA Development," Labcenter Electronics, [Online]. Available: https://www.labcenter.com/simulation/. [Accessed 11 Mei 2020].

[7] Jayant, "4x4 Matrix Keypad Interfacing with 8051 Microcontroller," 08 Agustus 2015. [Online]. Available: https://circuitdigest.com/microcontroller-projects/keypad-interfacing-with-8051-microcontroller. [Accessed 11 Mei 2020].

[8] Jameco, "How Do Servo Motors Work?," Jameco Electronics, [Online]. Available: https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html. [Accessed 11 Mei 2020].

[9] Admin, "Interfacing 16x2 LCD with 8051," Circuits Today, [Online]. Available: http://www.circuitstoday.com/interfacing-16x2-lcd-with-8051. [Accessed 11 Mei 2020].

[10] S. Khan, "Interfacing LED Dot Matrix with 8051 Microcontroller- (Part 5/45)," Engineers Garage, 25 Maret 2014. [Online]. Available: https://www.engineersgarage.com/contributions/interfacing-led-dot-matrix-with-8051-microcontroller-part-5-45/. [Accessed 11 Mei 2020].