BAB II

TINJAUAN PUSTAKA

2.1. Pengertian Banjir

Bencana banjir masih terjadi secara teratur dan terus-menerus di Indonesia. Banjir dapat terjadi akibat *volume* air yang berada di sungai melebihi badan sungai. Banyak dampak yang ditimbulkan oleh banjir, tidak hanya kerugian secara material, banjir juga dapat menimbulkan korban jiwa. Dampak dari banjir dapat dikurangi jika masyarakat lebih siap dalam menghadapi datangnya banjir tersebut. Salah satu caranya adalah dengan menyebarkan informasi *level* ketinggian air sungai secara cepat ke masyarakat.

Dalam cakupan pembicaraan yang luas, kita bisa melihat banjir sebagai suatu bagian dari siklus hidrologi, yaitu pada bagian air di permukaan Bumi yang bergerak ke laut. Dalam siklus hidrologi kita dapat melihat bahwa *volume* air yang mengalir di permukaan Bumi dominan ditentukan oleh tingkat curah hujan, dan tingkat peresapan air ke dalam tanah.

Aliran Permukaan = Curah Hujan – (Resapan ke dalam tanah + Penguapan ke udara)

Air hujan sampai di permukaan Bumi dan mengalir di permukaan Bumi, bergerak menuju ke laut dengan membentuk alur-alur sungai. Alur-alur sungai ini di mulai di daerah yang tertinggi di suatu kawasan, bisa daerah pegunungan, gunung atau perbukitan, dan berakhir di tepi pantai ketika aliran air masuk ke laut. Secara sederhana, segmen aliran sungai itu dapat kita bedakan menjadi

xdaerah hulu, tengah dan hilir. [3]

1. Daerah hulu: terdapat di daerah pegunungan, gunung atau perbukitan. Lembah sungai sempit dan potongan melintangnya berbentuk huruf "V". Di dalam alur sungai banyak batu yang berukuran besar (bongkah) dari runtuhan tebing, dan aliran air sungai mengalir di sela-sela batu-batu tersebut. Air sungai relatif sedikit. Tebing sungai sangat tinggi. Terjadi erosi pada arah vertikal yang dominan oleh aliran air sungai.
2. Daerah tengah: umumnya merupakan daerah kaki pegunungan, kaki gunung atau kaki bukit. Alur sungai melebar dan potongan melintangnya berbentuk huruf "U". Tebing sungai tinggi. Terjadi erosi pada arah horizontal, mengerosi batuan induk. Dasar alur sungai melebar, dan di dasar alur sungai terdapat endapan sungai yang berukuran butir kasar.
3. Daerah hilir: umumnya merupakan daerah dataran. Alur sungai lebar dan bisa sangat lebar dengan tebing sungai yang relatif sangat rendah dibandingkan lebar alur. Alur sungai dapat berkelok-kelok seperti huruf "S" yang dikenal sebagai "meander". Di kiri dan kanan alur terdapat dataran yang secara teratur akan tergenang oleh air sungai yang meluap, sehingga dikenal sebagai "dataran banjir". Di segmen ini terjadi pengendapan di kiri dan kanan alur sungai pada saat banjir yang menghasilkan dataran banjir. Terjadi erosi horizontal yang mengerosi endapan sungai itu sendiri yang diendapkan sebelumnya.

Dari karakter segmen-segmen aliran sungai itu, maka dapat dikatakan bahwa:

1. Banjir merupakan bagian proses pembentukan daratan oleh aliran sungai. Dengan banjir, sedimen diendapkan di atas daratan. Bila muatan sedimen sangat banyak, maka pembentukan daratan juga terjadi di laut di depan muara sungai yang dikenal sebagai "delta sungai."
2. Banjir yang meluas hanya terjadi di daerah hilir dari suatu aliran dan melanda dataran di kiri dan kanan aliran sungai. Di daerah tengah, banjir hanya terjadi di dalam alur sungai.

Dari pengertian di atas dapat disimpulkan bahwa banjir adalah peristiwa yang terjadi ketika aliran air yang berlebihan merendam daratan. Banjir juga dapat terjadi di sungai, ketika alirannya melebihi kapasitas saluran air, terutama di selokan sungai.

2.1.1. Jenis Banjir

Terdapat berbagai macam banjir yang disebabkan oleh beberapa hal, diantaranya **:**

1. Banjir air

Banjir yang satu ini adalah banjir yang sudah umum. Penyebab banjir ini adalah meluapnya air sungai, danau, atau selokan sehingga air akan meluber lalu menggenangi daratan. Umumnya banjir seperti ini disebabkan oleh hujan yang turun terus-menerus sehingga sungai atau danau tidak mampu lagi menampung air.

2. Banjir “Cileunang”

Jenis banjir yang satu ini hampir sama dengan banjir air. Namun banjir cileunang ini disebakan oleh hujan yang sangat deras dengan debit air yang sangat banyak. Banjir akhirnya terjadi karena air-air hujan yang melimpah ini tidak bisa segera mengalir melalui saluran atau selokan di sekitar rumah warga.

3. Banjir bandang

Tidak hanya banjir dengan materi air, tetapi banjir yang satu ini juga mengangkut material air berupa lumpur. Banjir seperti ini jelas lebih berbahaya daripada banjir air karena seseorang tidak akan mampu berenang ditengah-tengah banjir seperti ini untuk menyelamatkan diri. Banjir bandang mampu menghanyutkan apapun, karena itu daya rusaknya sangat tinggi.

4. Banjir rob (laut pasang)

Banjir rob adalah banjir yang disebabkan oleh pasangnya air laut. Banjir seperti ini kerap melanda kota Muara Baru di Jakarta. Air laut yang pasang ini umumnya akan menahan air sungai yang sudah menumpuk, akhirnya mampu menjebol tanggul dan menggenangi daratan.

5. Banjir lahar

Dingin Salah satu dari macam-macam banjir adalah banjir lahar dingin. Banjir jenis ini biasanya hanya terjadi ketika erupsi gunung berapi. Erupsi ini kemudian mengeluarkan lahar dingin dari puncak gunung.

2.2. *Buzzer*

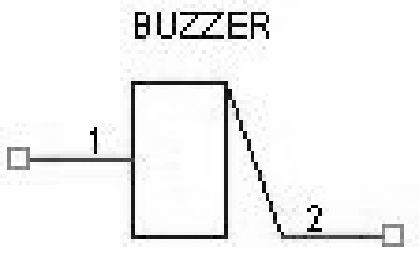
sebuah komponen elektronika yang berfungsi untuk mengubah getaran listrik menjadi getaran suara. Pada dasarnya prinsip kerja *buzzer* hampir sama

dengan *loud speaker*, jadi *buzzer* juga terdiri dari kumparan yang terpasang pada diafragma dan kemudian kumparan tersebut dialiri arus sehingga menjadi *elektromagnet*, kumparan tadi akan tertarik ke dalam atau keluar, tergantung dari arah arus dan polaritas magnetnya, karena kumparan dipasang pada diafragma maka setiap gerakan kumparan akan menggerakkan diafragma secara bolak-balik sehingga membuat udara bergetar yang akan menghasilkan suara. *Buzzer* biasa digunakan sebagai indikator bahwa proses telah selesai atau terjadi suatu kesalahan pada sebuah alat (*alarm*).[4]

Cara kerja *buzzer:*

pada saat aliran listrik atau tegangan listrik yang mengalir ke rangkaian yang menggunakan *piezoeletric* tersebut. *Piezo buzzer* dapat bekerja dengan baik dalam menghasilkan frekuensi di kisaran 1 - 6 kHz hingga 100 kHz.

Gambar 2.1 Bentuk Buzzer



Gambar 2.2 Simbol buzzer

2.3. Sensor Ultrasonik

Sensor ultrasonik adalah sebuah sensor yang memiliki fungsi untuk mengubah besaran fisis alias bunyi menjadi besaran listrik, begitupun sebaliknya. Prinsip kerja sensor ultrasonik ini cukup simpel, yakni berdasarkan pantulan suatu gelombang suara sehingga dapat digunakan untuk mendefiniskan eksistensi atau jarak suatu benda dengan frekuensi tertentu.

Gelombang ultrasonik sendiri memiliki frekuensi yang sangat tinggi, mencapai 20.000 Hz yang tidak bisa didengar oleh telinga manusia. Bunyi dengan frekuensi setinggi itu hanya bisa didengar oleh hewan-hewan tertentu seperti kucing, anjing, kelelawar, sampai dengan lumba-lumba.

Bunyi dari sensor ultrasonik sendiri dapat merambat melalui benda padat, cair, atau gas. Namun yang paling bagus adalah benda cair. Tak heran jika sensor yang satu ini banyak di aplikasikan pada kapal selam dan alat-alat khusus untuk mengukur kedalaman air laut. Sayangnya bunyi sensor ultrasonik dapat diserap oleh benda-benda tekstil dan busa. Sensor ini merupakan sensor ultrasonik siap pakai, satu alat yang berfungsi sebagai pengirim, penerima, dan pengontrol gelombang ultrasonik. Alat digunakan untuk mengukur jarak dari 2cm - 4m dengan akurasi 3mm. Alat ini memiliki 4 pin, pin *Vcc, Gnd, Trigger*, dan *Echo.* Pin *Vcc* untuk listrik positif dan Gnd untuk *ground*-nya. Pin *Trigger* untuk *trigger* keluarnya sinyal dari sensor dan pin *Echo* untuk menangkap sinyal pantul dari benda. Jarak benda dihitung berdasarkan rumus :

$$S = 340.t/2 \quad (2.1)$$

Keterangan rumus:

Dimana S merupakan jarak antara sensor ultrasonik dengan benda (bidang pantul), dan t adalah selisih antara waktu pemancaran gelombang oleh *transmitter* dan waktu ketika gelombang pantul diterima *receiver*. [5]

Gambar 2.3 Sensor Ultrasonik HC-SR04

2.4. Sensor *Passive Infrared* (PIR)

Sensor PIR (*Passive InfraRed*) adalah sensor yang digunakan untuk mendeteksi adanya pancaran sinar infra merah. Sensor PIR bersifat pasif, artinya sensor ini tidak memancarkan sinar infra merah tetapi hanya menerima radiasi sinar infra merah dari luar.

Sensor ini biasanya digunakan dalam perancangan *detektor* gerakan berbasis PIR. Karena semua benda memancarkan energi radiasi, sebuah gerakan akan terdeteksi ketika sumber infra merah dengan suhu tertentu (misal: manusia) melewati sumber infra merah yang lain dengan suhu yang berbeda (misal: dinding), maka sensor akan membandingkan pancaran infra merah yang diterima setiap satuan waktu, sehingga jika ada pergerakan maka akan terjadi perubahan pembacaan pada sensor.

Sensor PIR terdiri dari beberapa bagian yaitu:

1. *Fresnel Lens*

Lensa Fresnel pertama kali digunakan pada tahun 1980an. Digunakan sebagai lensa yang memfokuskan sinar pada lampu mercusuar. Penggunaan paling luas pada lensa *Fresnel* adalah pada lampu depan mobil, di mana mereka membiarkan berkas parallel secara kasar dari pemantul parabola dibentuk untuk memenuhi persyaratan pola sorotan utama. Namun kini, lensa *Fresnel* pada mobil telah ditiadakan diganti dengan lensa plain polikarbonat. Lensa *Fresnel* juga berguna dalam pembuatan film, tidak hanya karena kemampuannya untuk memfokuskan sinar terang, tetapi juga karena intensitas cahaya yang *relative* konstan diseluruh lebar berkas cahaya.

2. IR *Filter*

IR *Filter* dimodul sensor PIR ini mampu menyaring panjang gelombang sinar *infrared pasif* antara 8 sampai 14 mikrometer, sehingga panjang gelombang yang dihasilkan dari tubuh manusia yang berkisar antara 9 sampai 10 *mikrometer* ini saja yang dapat dideteksi oleh sensor. Sehingga Sensor PIR hanya bereaksi pada tubuh manusia saja.

3. *Pyroelectric Sensor*

Seperti tubuh manusia yang memiliki suhu tubuh kira-kira 32 derajat celcius, yang merupakan suhu panas yang khas yang terdapat pada lingkungan. Pancaran sinar inframerah inilah yang kemudian ditangkap oleh *Pyroelectric sensor* yang merupakan inti dari sensor PIR ini sehingga menyebabkan *Pyroelectic sensor* yang terdiri dari *galium nitrida, caesium*

nitrat dan *litium tantalate* menghasilkan arus listrik. Mengapa bisa menghasilkan arus listrik? Karena pancaran sinar inframerah pasif ini membawa energi panas. Material *pyroelectric* bereaksi menghasilkan arus listrik karena adanya energi panas yang dibawa oleh *infrared pasif* tersebut. Prosesnya hampir sama seperti arus listrik yang terbentuk ketika sinar matahari mengenai *solar cell*.

4. *Amplifier*

Sebuah sirkuit *amplifier* yang ada menguatkan arus yang masuk pada material *pyroelectric*.

5. *Komparator*

Setelah dikuatkan oleh *amplifier* kemudian arus dibandingkan oleh komparator sehingga mengahasilkan *output*.

Sensor PIR memiliki jangkauan jarak yang bervariasi, tergantung karakteristik sensor. Pada umumnya sensor PIR memiliki jangkauan pembacaan efektif hingga 5 meter, dan sensor ini sangat efektif digunakan sebagai *human detector*.[6]

Gambar 2.4 Sensor *Passive Infrared* (PIR)

2.5. *Microcontroler STM32F103C8T6*

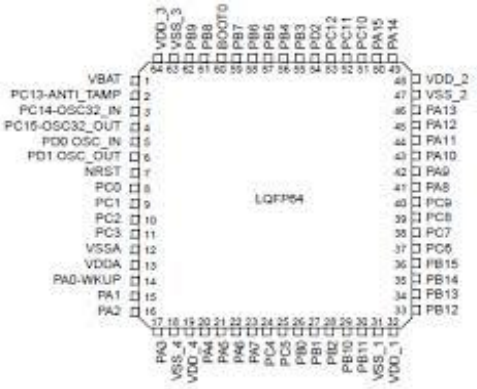
STM32 adalah keluarga dari 32-bit sirkuit terpadu mikrokontroler oleh ST*Microelectronics*. *Chip* STM32 dikelompokkan ke dalam seri terkait yang berbasis di sekitar inti prosesor ARM 32-bit yang sama, seperti *Cortex*-M7F ,*Cortex*-M4F , *Cortex*-M3 , *Cortex*-M0 + , atau *Cortex*-M0 . Secara *internal*, setiap mikrokontroler terdiri dari inti prosesor, memori RAM statis , memori *flash*, antarmuka *debugging*, dan berbagai perifera l [7].

2.5.1. Tinjauan MCU

Keluarga STM32F103xx *Microcontrollers* terdiri dari ARM *Cortex*-M3 32-bit RISC inti, memori tertanam berkecepatan tinggi (memori *Flash* hingga 128 Kbytes dan *Static Random Access Memory* (SRAM) hingga 20 Kbytes), I / Os (*Input / Output*) dan periferal yang mereka bekerja sama dengan menghubungkan ke dua bus APB (*Advanced Peripheral Bus*). Itu *Mikrokontroler* STM32F103xx mencakup banyak periferal serta dua ADC 12 bit, sebuah *Advanced Control Timer*, tiga *timer* 16-bit *General Purpose* dan juga PWM (*Pulse Width* Modulasi) pengatur waktu. Ini juga disediakan oleh dua I²Cs (*Inter-Integrated Circuit)* dan SPI (*Serial* Antarmuka Periferal), tiga USART (*Universal Synchronous / Asynchronous Receive Transmitter*), USB dan CAN (*Controller Area Network*) sebagai antarmuka komunikasi sistem.

Mikrokontroler berdensitas menengah digunakan dan memiliki 64 pin. Keluarga *mikrokontroler* ini terdiri dari tiga port yang PA, PB dan PC adalah port MCU dan masing-masing port memiliki 16 pin sebagai I / Os. V SS , V DD dan V BAT digunakan untuk bias *mikrokontroler* dengan menggunakan *catu* daya

eksternal. Gambar 2.5 menyajikan *pinout* untuk keluarga *STM32F103* yang digunakan dalam penelitihan ini.



Gambar 2.5. *Pinout* Kinerja *STM32F103*.

2.5.2. Kelompok Keluarga STM32F103

Mikrokontroler keluarga STM32F103xx dibagi menjadi tiga kelompok:

- **Kerapatan rendah:** STM32F103x4 dan STM32F103x6 adalah perangkat berdensitas rendah.
- **Kerapatan sedang:** STM32F103x8 dan STM32F103xB adalah Kerapatan sedang perangkat.
- ***High-density*:** STM32F103xC, STM32F103xD dan STM32F103xE adalah *High*-perangkat kepadatan.

Pinout	Low-density devices		Medium-density devices		High-density devices		
	16 KB Flash	32 KB Flash(1)	64 KB Flash	128 KB Flash	256 KB Flash	384 KB Flash	512 KB Flash
	6 KB RAM	10 KB RAM	20 KB RAM	20 KB RAM	48 KB RAM	64 KB RAM	64 KB RAM
144					5 × USARTs 4 × 16-bit timers, 2 × basic timers 3 × SPIs, 2 × I^2Ss, 2 × I2Cs USB, CAN, 2 × PWM timers 3 × ADCs, 1 × DAC, 1 × SDIO FSMC (100 and 144 pins)		
100			3 × USARTs 3 × 16-bit timers 2 × SPIs, 2 × I^2Cs, USB, CAN, 1 × PWM timer 2 × ADC				
64	2 × USARTs 2 × 16-bit timers 1 × SPI, 1 × I^2C, USB, CAN, 1 × PWM timer 2 × ADCs						
48							
36							

Gambar 2.6. Keluarga STM32F103

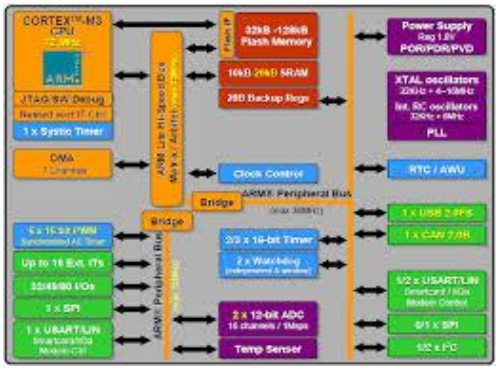
Ketiga kelompok ini dibuat sesuai dengan fitur *mikrokontroler* STM32F103xx anggota keluarga. *Mikrokontroler* berdensitas rendah memiliki memori *Flash* dan RAM yang lebih rendah (Akses Acak Memori), timer kurang dan periferal di bandingkan dengan dua kelompok lainnya. Dengan demikian, Menengah- kepadatan dan *High-density* terdiri dari memori *Flash* yang lebih tinggi, kapasitas *RAM* dan juga lebih banyak periferal tambahan. Keluarga dengan kepadatan rendah mencakup 16 KB hingga 32 KB memori *Flash* dan 6 KB hingga 10 KB *RAM* kapasitas. Mereka terdiri dari 1 × BISA, 1 × *USB*, 1 × *PWM timer*, 1 × I²C, 1 × SPI dan 2 × *ADC*, 2 × *USART*, dan 2 × 16-bit timer.

Perbedaan antara keluarga berkepadatan rendah berkenaan dengan jumlah paket *pinout* mereka. Ada tiga jenis paket *pinout* yang mereka buat 36, 48 dan 64 pin. Memori *Flash* mereka ditingkatkan dari 64 KB menjadi 128 Kapasitas KB dan RAM adalah 20 KB. Keluarga dengan kepadatan menengah memiliki lebih banyak *properti* dibandingkan dengan keluarga dengan kepadatan rendah. Itu jumlah periferal dan pinouts dari mereka ditingkatkan. Mereka memiliki 1 × BISA, 1 × USB, 1 × *timer PWM*, 2 × I²C, 2 × SPI, 2 × ADC, 3 × USART, dan 3 × 16-*bit timer*. Mereka juga memiliki tiga jenis paket *pinout* yang terdiri dari 48, 64 dan 100 pin. Keluarga dengan kepadatan tinggi diselesaikan lebih dari yang lain dan memiliki lebih banyak periferal. Mereka membuat up periferal ini, seperti 1 × BISA, 1 × USB, 1 × *PWM timer*, 2 × I²Ss (SPI), 2 × I²C, 3 × SPI, 2 × ADC, 1 × DAC, 5 × USART, 2 × *timer* dasar, 4 × 16-bit *timer*, 1 × SDIO (*Output Input Digital* Aman), dan 1 × FSMC (Pengontrol Memori *Statis Fleksibel*).

Untuk proyek ini *mikrokontroler* STM32F103xx (perangkat Kerapatan Sedang) diterapkan dan harus disebutkan bahwa semua lini kinerja keluarga STM32 sepenuhnya *kompatibel*. Gambar 2.6. memperkenalkan secara singkat semua fitur untuk *mikrokontroler* STM32F103xx secara singkat. Ini tiga kategori untuk *mikrokontroler* STM32F103xx berguna untuk *desainer* yang mereka dapat pilih keluarga yang cocok untuk proyek mereka. Kebenaran besar dari STM32 ini menyebabkan berkurangnya harga produk dan juga produk dapat memiliki bentuk dan ukuran yang lebih baik. *Mikrokontroler* STM32F103xx sangat nyaman digunakan untuk berbagai jenis aplikasi, serta, Industri, *Kontrol*, Kedokteran, PC *peripheral gaming*, GPS *(Global* Sistem Pemosisian), *intercom Video*, Sistem *Alarm*, dll.

2.5.3. Komponen *MCU STM32F103*

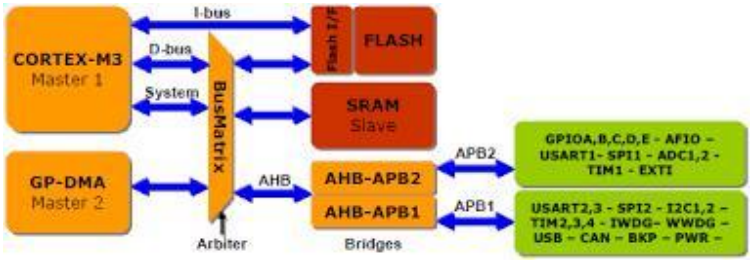
Bagian ini menjelaskan inti *mikrokontroler*, memori, I / Os dan periferal yang diperkenalkan pada gambar 2.7. mengenai *diagram blok* kinerja. Ada penjelasan singkat untuk STM32F103 bagian mikrokontroler di bawah ini.



Gambar 2.7. *Diagram Blok* Kinerja STM32F103.

2.5.4. *Arsitektur Sistem*

Terdiri dari Bus, dan DMA tujuan Umum (Memori Langsung Akses), SRAM *Internal*, Memori *Flash Internal* yang sebagian dari mereka anggap sebagai *master* dan yang lain anggap sebagai pekerja. Gambar 2.8. ditunjukkan *arsitektur* bus untuk *mikrokontroler* keluarga STM32. Itu akses antara bus sistem inti dan bus DMA dikendalikan oleh *BusMatrix*.



Gambar 2.8. *Arsitektur* Sistem.

2.5.5. Inti *ARM Cortex-M3*

Adalah ***CPU*** *mikrokontroler* dan merupakan salah satu yang paling signifikan bagian dari mikrokontroler. Inti ini adalah versi terakhir dari prosesor ARM yang diterapkan untuk sistem embedded. Ini memiliki spesifikasi yang baik seperti frekuensi *maksimum* 72 MHz, 90DIMPS (*Distributed Integrated Message Processing System*) dengan 1,25 DIMPS / MHz, kinerja pada akses memori nol negara, *Single-siklus* perkalian dan pembagian perangkat keras, Pengontrol interupsi bersarang (saluran 43 *interrupt maskable*, pemprosesan *interrupt*), *Low-*konsumsi daya, harga rendah, jumlah gerbang rendah, dll.

2.5.6. Sistem Memori

Mikrokontroler ini terdiri dari dua bagian yaitu *Flash* memori dan SRAM (*Static Random Access Memory*) memori. Memori *flash* untuk menyimpan data

dan program dan kapasitasnya hingga 128 Kbytes. Memori SRAM adalah untuk membaca / menulis pada CPU dengan status tunggu nol untuk menyimpan data untuk diproses oleh USB dan kapasitasnya sudah habis hingga 20 Kbytes.

2.5.7. *Nested Vect It Ctrl (NVIC)*

Nested Vect It Ctrl (NVIC) adalah singkatan dari *Nested Vector Interrupt Controller*. Itu bisa digunakan untuk mengontrol hingga 43 saluran interupsi maskable (maskable *interrupt* adalah interupsi khusus yang dapat diaktifkan / di*nonaktifkan* atau dikelola oleh *programmer)*, dan itu memiliki 16 *programmable* tingkat prioritas. Ini digunakan untuk mengatur prioritas saluran IRQ (*Interrupt Request*).

2.5.8. *Ext. ITS* (*EXTI*)

Ext. ITS (EXTI) adalah singkatan dari *External Interrupt / Event Controller* itu memiliki sembilan belas ujung garis *detektor* untuk membuat permintaan interupsi / acara. Untuk mendeteksi pemicu eksternal dapat digunakan, dapat dipicu oleh naik, jatuh atau keduanya memicu dan itu juga dapat digunakan. Ini mungkin untuk dihubungkan hingga delapan puluh GPIO hingga enam belas garis interupsi eksternal untuk dideteksi pemicu *eksternal* (sebagai interupsi).

2.5.9. Sistem Jam (SYSCLK)

Sistem Jam (SYSCLK) Terdiri dari sumber *clock* berbeda di *mikrokontroler* sebagai baik seperti: HSI (*High Speed Internal*) *osilator clock*, HSE (*High Speed External*) osilator jam, PLL (*Phase Locked Loop*) jam, dan LSI RC (*Low Speed Internal Resistor dan Oscillator Capasitor*) dan LSE (*Low Speed*

External) *Osilator*. Jam *osilator* HSI adalah sinyal jam Kecepatan Tinggi Internal yang merupakan RC *internal* (8 MHz) *osilator*. Itu selalu diterapkan sebagai sistem jam untuk MCU secara default. HSE adalah sinyal jam Ekstra Kecepatan Tinggi dan dapat menggunakan *osilator* eksternal 4 hingga 16 MHz di perintah untuk menghasilkan sinyal *clock* yang sangat akurat untuk menyiapkan sistem jam. PLL dapat menghasilkan sinyal output yang akurat dan stabil dari frekuensi rendah yang tetap. Itu melipatgandakan output HSI RC atau frekuensi *clock output* kristal HSE untuk membuat frekuensi yang berbeda untuk prosesor *mikrokontroler* (CPU) dan periferal yang dipilih.

Jam LSI RC adalah sinyal *clock Internal* Kecepatan Rendah dan frekuensi *clock*-nya sekitar 40 KHz. Jam RC LSI dapat digunakan untuk sumber *clock* daya rendah dan berguna dalam *Stop* atau Modus siaga. LSE *Oscillator* adalah kristal eksternal dengan Kecepatan Rendah, frekuensinya 32.768 KHz dan siap digunakan Jam Waktu Nyata. Sumber-sumber jam ini menyediakan jam frekuensi untuk setiap bagian dari mikrokontroler seperti CPU, periferal, dll.

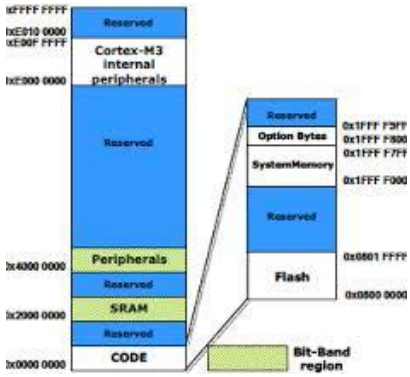
2.5.10. Jam *Startup*

Jam *Startup* adalah sistem jam untuk *mikrokontroler* saat *start up* (ketika MCU mulai kerja). Jam startup dibuat oleh HSI RC secara *default* dan jam ini 8 MHz karena HSI.

2.5.11. *Mode Boot*

Menentukan cara mem-*boot* CPU agar berfungsi. Ada tiga *mode boot* untuk mikrokontroler ini. Peta memori mikrokontroler ditunjukkan pada gambar

2.9. dimungkinkan untuk menggunakan bagian yang berbeda dari memori *mikrokontroler* untuk dapat mem-boot CPU.



Gambar 2.9. Peta Memori STM32F103.

Salah satu *mode boot* yang diperkenalkan di bawah ini dapat digunakan di *Startup* untuk *boot* CPU,

- *Flash* Pengguna: CPU akan melakukan *boot* dari *Flash* Pengguna.
- Memori Sistem: CPU akan *boot* dari memori Sistem.
- SRAM: CPU akan *boot* dari SRAM.

Sebuah *boot loader* diperlukan untuk memprogram atau memprogram ulang memori Flash dengan menggunakan USART. *Boot loader* ini ditempatkan di memori sistem.

2.5.12. *Power Supply*

Adalah untuk bias *mikrokontroler* untuk menyiapkan daya listrik yang cukup untuk berbagai bagian sistem. Dijelaskan pin yang harus terhubung ke *catu* daya di berikut.

- V DD : Pin ini menyiapkan *catu* daya untuk I / Os dan *Voltage Regulator* (*internal*). V DD harus terhubung ke 2,0 hingga 3,6 *Volt*e dan

tegangan ini diberikan secara *eksternal* melalui VDD pin pada MCU.

- V SSA , V DDA : Pin ini menyiapkan *catu* daya *analog eksternal* untuk ADC, *Reset blok*, RC dan PLL. Pin ini (V SSA , V DDA) harus dihubungkan ke 2,0 hingga 3,6 *Voltase*.
- V BAT : Pin ini menyiapkan *catu* daya untuk RTC (Jam Waktu Nyata), Jam *Internal osilator* (32 KHz) dan daftar Cadangan.

Mikrokontroler ini terdiri dari POR (*Power On Reset*) / PDR (*Power Down Reset*), itu adalah sebuah *Sirkuit* Terpadu dan selalu diaktifkan untuk memperbaiki operasi mulai dari 2,0 *Volt*. Ketika VDD kurang dari ambang spesifik (V POR / PDR), *mikrokontroler* tidak dapat berfungsi dan itu tetap pada suasana ulang. PVD (*Programmable Voltage Detector*) membandingkan V DD dengan V PVD ambang. *Interupsi servis* rutin menginformasikan *mikrokontroler* atau memasukkannya ke keadaan aman, jika V DD *voltage goes* naik atau turun V PVD (tegangan ambang).

2.5.13. Akses Memori Langsung (DMA)

Men*transfer* data dari Memori ke Memori, Memori ke *Peripheral* dan *Peripheral to Memory*. DMA terdiri dari tujuh saluran (setiap saluran digunakan oleh perangkat keras yang diminta DMA) dan itu dapat diterapkan dengan *periferal* utama seperti ADC (*Analog to Digital Converter*), TIMx (Tujuan Umum dan Kontrol Lanjut Pengatur Waktu), SPI, USART, I^2C , dll.

2.5.14. *Real Time Clock (RTC)*

register Backup menyiapkan program 32 bit yang dapat deprogram *counter* untuk MCU ini untuk menyediakan satu set konter yang terus

berjalan. Mereka diterapkan untuk menyiapkan fungsi Kalender Jam, *interupsi* berkala, dan *interupsi alarm*. Daya disediakan oleh pin V DD atau V BAT untuk *register* RTC dan *Backup*.

2.5.15. Pengatur Waktu Umum (TIMx)

Membentuk tiga pengatur waktu standar yang mampu disinkronkan agar diterapkan untuk MCU. Masing-masing *timer* ini terdiri dari 16 bit *counter* (*auto reloadable*), *output mode* pulsa atau PWM (*Pulse Width Modulation*), dan mereka juga punya *prescaler* 16 bit dan empat saluran *independen* untuk menangkap *input / output* atau membandingkan.

2.5.16. Pengatur Waktu Kontrol Lanjut (TIM1)

Sama seperti TIMx jika dikonfigurasi sebagai 16 bit *timer standar*, tetapi lebih lengkap. Dapat menggunakan empat saluran *independen* untuk *Input capture*, *Output* membandingkan, generasi PWM, dll. Ini juga dapat bekerja sebagai PWM tiga *fase multiplexing* pada enam saluran.

2.5.17. *Inter-Integrated Circuit* (I²C)

Adalah antarmuka bus dan dapat melakukan dalam dua mode yang mode ini adalah budak dan tuan. *Mikrokontroler* ini terdiri dari dua bus **I²C** antarmuka.

2.5.18. *Universal Synchronous / Asynchronous Receive Transmitter* (USART)

Adalah *port serial* dan ada dua USART tersedia. Salah satunya dapat berkomunikasi hingga 4,5 Mbit / S dan satu lagi hingga 2,25 Mbit / S.

2.5.19. *Serial Peripheral Interface* (SPI)

Adalah *port serial* dan dapat berkomunikasi di kecepatan hingga 18 Mbit / S. Ada dua SPI tersedia yang dapat menggunakan pengontrol DMA.

2.5.20. *Controller Area Network* (CAN)

Adalah bus standar yang digunakan untuk membuatnya komunikasi antara *mikrokontroler* dan perangkat tanpa menggunakan *host* (komputer) apa pun. Saya dapat bertransaksi dengan kecepatan 1 Mbit / S.

2.5.21. *Universal Serial Bus* (USB)

Adalah *bus serial* yang digunakan untuk mentransaksikan data antara MCU dan PC (*host*). Ada empat jenis USB yang dikelompokkan sesuai kecepatan mereka.

▪ Kecepatan Rendah: Kecepatannya adalah 1,5 Mbit / S

▪ Kecepatan Penuh: Lajunya adalah 12 Mbit / S

▪ *Hi Speed*: Kecepatannya 480 Mbit / S

▪ Kecepatan Super: Rasionya 5 Gbit / S

Perangkat USB untuk MCU ini adalah versi Kecepatan Penuh (12 Mbit / S).

2.5.22. *Input / Output* Tujuan Umum (GPIO)

Adalah antarmuka untuk MCU untuk hubungkan ke perangkat *eksternal* yang dapat digunakan sebagai *input* untuk membaca data atau gunakan sebagai *output* untuk ditulis data, dll. Tersedia tiga GPIO pada MCU ini (STM32F103 dengan 80 Pin). Mereka harus dikonfigurasi oleh perangkat lunak dan mereka signifikan untuk dipertimbangkan selama bekerja dengan MCU. Semua perangkat MCU dapat terhubung ke luar atau perangkat *eksternal* lainnya dengan GPIO.

2.5.23. *Analog ke Digital Konverter* (ADC)

Mengubah sinyal *analog input kontinu* ke nilai digit untuk digunakan oleh perangkat *digital* seperti MCU, PC, dll. *Elektronik* ini perangkat menerima

tegangan *analog* atau arus sebagai *input* untuk menciptakan nilai *diskrit* pada *output*nya. *Mikrokontroler* ini memiliki dua ADC yang resolusinya adalah 12 bit dan masing-masing satu memiliki 16 saluran [8].

2.6. *ST-LINK V2*

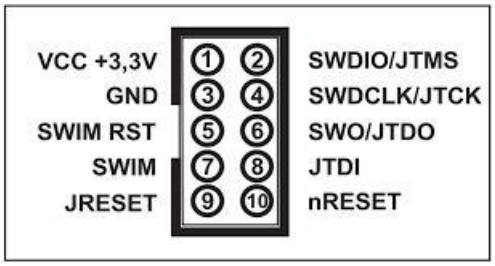
ST-Link V2 ini merupakan *ST-Link* V2 versi mini yang dapat memprogram seri MCU STM dari ST*Microelectronic* STM8 dan STM32, simulasi dan *synchronous debugging.* Sangat mudah digunakan dan dibawa kemana-mana karena ukurannya lebih besar sedikit dari jempol dengan harga jauh lebih murah dan dan kinerjanya setara dengan ST-*Link* V2 buatan *STMicroelectronic* dan dilindungi dengan *metal case* dan penutup sehingga aman dari benturan atau yang lainnya.

Gambar 2.10. *ST-LINK V2 Mini.*

Fungsi *ST-Link* V2:

1. Fungsi pemrograman : Mendukung *FLASH ROM / EEPROM / AFR programming*.

2. Kinerja Simulasi: Simulasi dan *debugging* menggunakan konektor USB2.0, Cepat dalam kecepatan.
3. Kinerja Pemrograman: *Download* kode menggunakan *konektor* USB2.0 dalam mode SWIM untuk STM8 dan mode SWD untuk STM32, *download* cepat hanya beberapa detik saja.
4. *Firmware update*: *Firmware* dapat diperbaharui. dapat diperbaharui secara otomatis untuk mengaptasi dengan produk ST*Microelectronic* sehingga mudah digunakan.



Gambar 2.11. *Pinout ST-Link.*

Software yang dapat digunakan untuk mini *ST-Link*/V2 ini:

a. *ST-Link Utility* di atas 2.0
b. *STVD* di atas 4.2.1
c. *STVP* adi atas 3.2.3
d. *IAR WERAM* di atas V6.20
e. *IAR EWSTM8* di atas V1.3
f. *KEIL RVMDK* di atas V4.21

2.7. *Liquid Crystal Display* (LCD) 16 x 2

LCD (*Liquid Crystal Display*) adalah suatu jenis media tampil yang menggunakan kristal cair sebagai penampil utama. LCD sudah digunakan

diberbagai bidang misalnya alal–alat elektronik seperti televisi, kalkulator, atau pun layar komputer. Pada postingan aplikasi LCD yang dugunakan ialah LCD *dot matrik* dengan jumlah karakter 2 x 16. LCD sangat berfungsi sebagai penampil yang nantinya akan digunakan untuk menampilkan status kerja alat.

Adapun fitur yang disajikan dalam LCD ini adalah :

a. Terdiri dari 16 karakter dan 2 baris.

b. Mempunyai 192 karakter tersimpan.

c. Terdapat karakter generator terprogram.

d. Dapat dialamati dengan mode 4-bit dan 8-bit.

e. Dilengkapi dengan *back light*.

Gambar 2.12. Bentuk Fisik LCD 16 x 2

Spesifikasi Kaki LCD 16 x 2

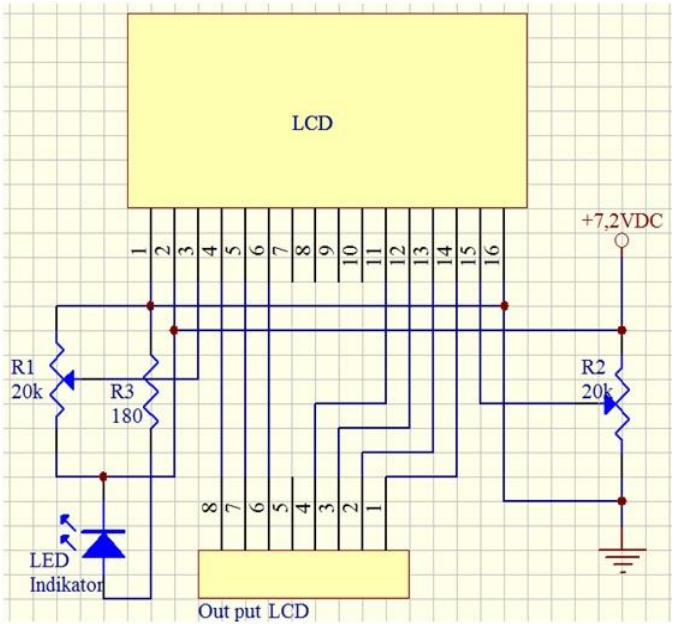
Pin	*Deskripsi*
1	*Ground*
2	*Vcc*
3	Pengatur kontras
4	*"RS" Instruction/Register Select*
5	*"R/W" Read/Write LCD Registers*

6	*"EN" Enable*
7-14	Data I/O Pins
15	Vcc
16	*Ground*

Cara Kerja LCD Secara Umum Pada aplikasi umumnya RW diberi logika rendah "0". Bus data terdiri dari 4-biT atau 8-bit. Jika jalur data 4-bit maka yang digunakan adalah DB4 sampai dengan DB7. Sebagaimana terlihat pada *table diskripsi, interface* LCD merupakan sebuah *parallel* bus, dimana hal ini sangat memudahkan dan sangat cepat dalam pembacaan dan penulisan data dari atau ke LCD. Kode ASCII yang ditampilkan sepanjang 8-bit dikirim ke LCD secara 4-bit atau 8 bit pada satu waktu. Jika mode 4-bit yang digunakan, maka 2 nibble data dikirim untuk membuat sepenuhnya 8-bit (pertama dikirim 4-bit MSB lalu 4-bit LSB dengan pulsa *clock* EN setiap *nibble*nya). Jalur *kontrol* EN digunakan untuk memberitahu LCD bahwa *mikrokontroller* mengirimkan data ke LCD. Untuk mengirim data ke LCD program harus menset EN ke kondisi *high* "1" dan kemudian menset dua jalur kontrol lainnya (RS dan R/W) atau juga mengirimkan Data.

Saat jalur lainnya sudah siap, EN harus diset ke "0" dan tunggu beberapa saat (tergantung pada datasheet LCD), dan set EN kembali ke *high* "1". Ketika jalur RS berada dalam kondisi *low* "0", data yang dikirimkan ke LCD dianggap sebagai sebuah perintah atau instruksi khusus (seperti bersihkan layar, posisi *kursor* dll). Ketika RS dalam kondisi *high* atau "1", data yang dikirimkan adalah data ASCII yang akan ditampilkan dilayar. Misal, untuk menampilkan huruf "A"

pada layar maka RS harus diset ke “1”. Jalur *kontrol* R/W harus berada dalam kondisi *low* (0) saat informasi pada data bus akan dituliskan ke LCD. Apabila R/W berada dalam kondisi *high* “1”, maka program akan melakukan *query* (pembacaan) data dari LCD. Instruksi pembacaan hanya satu, yaitu *Get LCD* status (membaca status LCD), lainnya merupakan instruksi penulisan. Jadi hampir setiap aplikasi yang menggunakan LCD, R/W selalu diset ke “0”. Jalur data dapat terdiri 4 atau 8 jalur (tergantung mode yang dipilih pengguna), DB0, DB1, DB2, DB3, DB4, DB5, DB6 dan DB7. Mengirim data secara *parallel* baik 4-bit atau 8-bit merupakan 2 mode operasi *primer*. Untuk membuat sebuah aplikasi *interface* LCD, menentukan mode operasi merupakan hal yang paling penting.



Gambar 2.13. Skematik LCD 16x2.

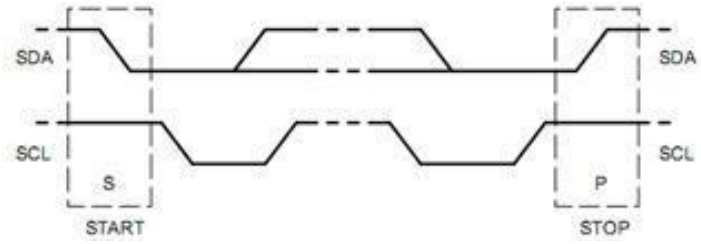
Mode 8-bit sangat baik digunakan ketika kecepatan menjadi keutamaan dalam sebuah aplikasi dan setidaknya minimal tersedia 11 pin I/O (3 pin untuk *kontrol*, 8 pin untuk data). Sedangkan mode 4 bit minimal hanya membutuhkan 7-

bit (3 pin untuk kontrol, 4 pin untuk data). Bit RS digunakan untuk memilih apakah data atau instruksi yang akan di*transfer* antara *mikrokontroller* dan LCD. Jika bit ini di set (RS = 1), maka *byte* pada posisi kursor LCD saat itu dapat dibaca atau ditulis. Jika bit ini di *reset* (RS = 0), merupakan instruksi yang dikirim ke LCD atau status eksekusi dari instruksi terakhir yang dibaca. Untuk gambar skematik LCD 16x2 adalah sebagai berikut.[9]

2.8. I2C

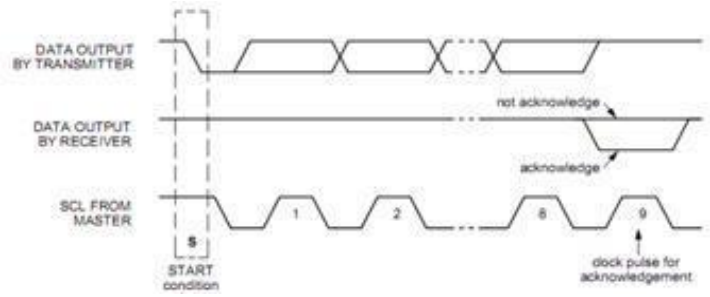
Inter Integrated Circuit atau sering disebut I^2C adalah standar komunikasi *serial* dua arah menggunakan dua saluran yang didisain khusus untuk mengirim maupun menerima data. Sistem I^2C terdiri dari saluran SCL (*Serial Clock*) dan SDA (*Serial Data*) yang membawa informasi data antara I^2C dengan pengontrolnya. Piranti yang dihubungkan dengan sistem I2C Bus dapat dioperasikan sebagai *Master* dan *Slave*. *Master* adalah piranti yang memulai *transfer* data pada I^2C Bus dengan membentuk sinyal *Start*, mengakhiri *transfer* data dengan membentuk sinyal *Stop*, dan membangkitkan sinyal *clock*. *Slave* adalah piranti yang dialamati *master.*

Sinyal *Start* merupakan sinyal untuk memulai semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "1" menjadi "0" pada saat SCL "1". Sinyal *Stop* merupakan sinyal untuk mengakhiri semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "0" menjadi "1" pada saat SCL "1". Kondisi sinyal *Start* dan sinyal *Stop* seperti tampak pada Gambar 2.14.



Gambar 2.14. Kondisi sinyal *start* dan *stop*.

Sinyal dasar yang lain dalam I2C Bus adalah sinyal *acknowledge* yang disimbolkan dengan ACK Setelah *transfer* data oleh master berhasil diterima *slave, slave* akan menjawabnya dengan mengirim sinyal *acknowledge*, yaitu dengan membuat SDA menjadi "0" selama *siklus clock* ke 9. Ini menunjukkan bahwa *Slave* telah menerima 8 bit data dari Master. Kondisi sinyal acknowledge seperti tampak pada Gambar 2.15.

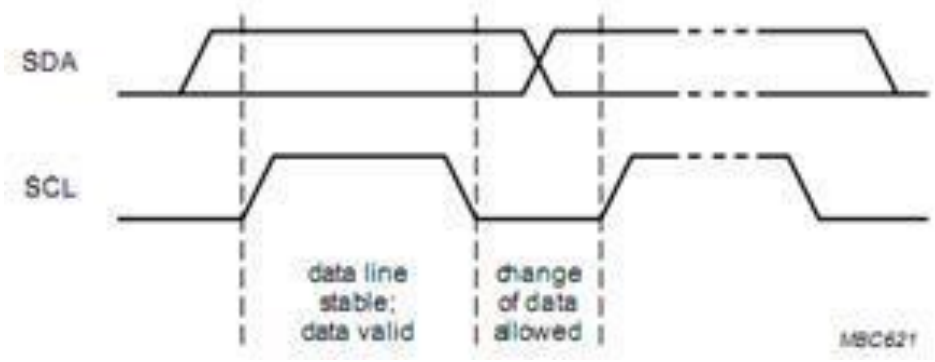


Gambar 2.15. Sinyal ACK dan NACK.

Dalam melakukan *transfer* data pada I2C Bus, kita harus mengikuti tata cara yang telah ditetapkan yaitu:

- Transfer data hanya dapat dilakukan ketikan Bus tidak dalam keadaan sibuk.
- Selama proses *transfer* data, keadaan data pada SDA harus stabil selama SCL dalam keadan tinggi. Keadaan perubahan "1" atau "0" pada SDA

hanya dapat dilakukan selama SCL dalam keadaan rendah. Jika terjadi perubahan keadaan SDA pada saat SCL dalam keadaan tinggi, maka perubahan itu dianggap sebagai sinyal *Start* atau sinyal *Stop* [10].



Gambar 2.16. *Trasfer* Bit pada I^2C bus.

2.9. *Module NodeMCU ESP8266*

NodeMCU adalah platform IoT *open source*. Ini termasuk *firmware* yang berjalan di ESP8266 *Wi-Fi* SoC dari *Espressif Systems*, dan perangkat keras yang berbasis pada modul ESP-12. Istilah "NodeMCU" secara *default* mengacu pada *firmware* daripada perangkat dev. *Firmware* menggunakan bahasa *scripting Lua*. Hal ini didasarkan pada proyek eLua, dan dibangun di atas *SDK Non-OS Espresso* untuk ESP8266. Ini menggunakan banyak proyek *open source*, seperti lua-cjson, dan spiffs.

NodeMCU diciptakan tak lama setelah ESP8266 keluar. Pada tanggal 30 Desember 2013, *Espressif Systems* memulai produksi ESP8266. ESP8266 adalah *Wi-Fi* SoC yang terintegrasi dengan inti *Tensilica Xtensa* LX106, banyak digunakan dalam aplikasi IoT. NodeMCU dimulai pada 13 Okt 2014, ketika Hong melakukan file pertama *firmware* nodemcu ke *GitHub*. Dua bulan kemudian,

proyek tersebut diperluas untuk menyertakan *open-hardware platform* saat pengembang Huang R membuat file *gerber* dari papan ESP8266, yang diberi nama devkit v0.9. Belakangan bulan itu, Tuan PM mem-*porting client library* MQTT (*Message Queue Telemetry Transport*) dari *Contiki* ke *platform* SoC ESP8266, dan berkomitmen pada proyek NodeMCU, kemudian NodeMCU bisa mendukung protokol MQTT IoT, menggunakan Lua untuk mengakses *broker* MQTT. Pembaruan penting lainnya dilakukan pada tanggal 30 Januari 2015, saat Devsaurus mem-*porting* proyek u8glib ke NodeMCU, memudahkan NodeMCU untuk menggerakkan LCD, Layar, OLED, bahkan *display* VGA. Di musim panas 2015 para pencipta meninggalkan proyek *firmware* ini dan sekelompok kontributor *independen* namun berdedikasi mengambil alihnya. Pada musim panas 2016, NodeMCU menyertakan lebih dari 40 modul yang berbeda. Karena keterbatasan sumber daya, pengguna perlu memilih modul yang *relevan* untuk proyek mereka dan membuat *firmware* yang disesuaikan dengan kebutuhan. Beberapa fitur dari NodeMCU anatara lain:

- Perangkat keras IO (*input-output*) seperti Arduino API tingkat lanjut untuk perangkat keras IO, yang dapat secara dramatis mengurangi pekerjaan yang berlebihan untuk mengkonfigurasi dan memanipulasi perangkat keras. Kode seperti arduino, tapi *interaktif* dalam naskah Lua.
- API (*application programming interface*) jaringan dengan gaya *Nodejs Event-driven API* untuk aplikasi jaringan, yang memudahkan pengembang menulis kode yang berjalan pada MCU *(memory controller unit)* berukuran 5mm*5mm dengan gaya Nodejs, dan sangat mempercepat aplikasi pengembangan aplikasi IOT.

- Perangkat *WI-FI* paling murah Perangkat *WIFI* MCU ESP8266 terintegrasi dan mudah untuk *prototyping development kit* dengan harga kurang dari $2 [11].

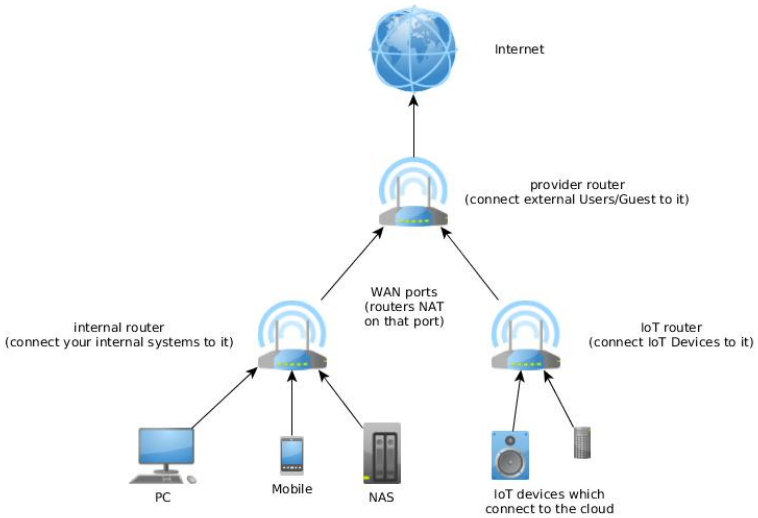
Gambar 2.17. Bentuk Fisik NodeMCU ESP8266.

2.10. *Internet of Things* (IOT)

Internet of Things (IOT) adalah struktur di mana objek, orang disediakan dengan identitas eksklusif dan kemampuan untuk pindah data melalui jaringan tanpa memerlukan dua arah antara manusia ke manusia yaitu sumber ke tujuan atau interaksi manusia ke *computer* [12]. *Internet of Things* merupakan perkembangan keilmuan yang sangat menjanjikan untuk mengoptimalkan kehidupan berdasarkan sensor cerdas dan peralatan pintar yang bekerjasama melalui jaringan *internet* [13].

Pada tahun 2008 FCC menyetujui penggunaan "*white space spectrum".* Akhirnya peluncuran IPv6 di tahun 2011 memicu pertumbuhan besar di bidang Internet of Things, perkembangan ini didukung oleh perusahaan raksasa seperti Cisco, IBM, Ericson mengambil inisiatif banyak dari pendidikan dan komersial dengan IOT teknologi dapat hanya dijelaskan sebagai hubungan antara manusia dan komputer. Perkembangan *Internet of Things*, semua peralatan yang digunakan dalam kehidupan sehari hari dapat dikendalikan

dan dipantau menggunakan IOT. Mayoritas proses dilakukan dengan bantuan sensor di IOT. Sensor dikerahkan di mana mana dan sensor ini meng*konversi* data fisik mentah menjadi sinyal digital dan mengirimkan mereka ke pusat kontrol. Dengan cara ini dapat me*monitor* perubahan lingkungan jarak jauh dari setiap bagian dari dunia melalui i*nternet*. *Arsitektur* sistem ini akan didasarkan pada *konteks* operasi dan proses dalam skenario *real-time*. Di otomasi rumah setiap kotak saklar listrik akan terhubung dengan ponsel pintar (atau kadang-kadang *remote*) sehingga itu bisa dioperasikan dari jarak jauh. Tapi skenario seperti itu tidak perlu *prosesor* dan perangkat penyimpanan dipasang di setiap kotak *saklar*. Hanya dibutuhkan sensor untuk menangkap sinyal dan proses itu (kebanyakan beralih *ON / OFF*). Jadi *arsitektur* sistem ini bervariasi tergantung pada *konteks* penerapannya [14].



Gambar 2.18. *Ilustrasi Internet of Things.*

2.11. *BLYNK*

Blynk adalah sebuah layanan *server* yang digunakan untuk mendukung *project Internet of Things.* Layanan *server* ini memiliki lingkungan *mobile user*

baik Android maupun iOS. Blynk Aplikasi sebagai pendukung IoT dapat diundung melalui *Google play*. Blynk mendukung berbagai macam *hardware* yang dapat digunakan untuk *project Internet of Things*. Blynk adalah *dashborad* digital dengan fasilitas antarmuka grafis dalam pembuatan *project*nya. Penambahan komponen pada Blynk Apps dengan cara *Drag and Drop* sehingga memudahkan dalam penambahan komponen *Input/output* tanpa perlu kemampuan pemrograman Android maupun iOS.

Blynk diciptakan dengan tujuan untuk *control dan monitoring hardware* secara jarak jauh menggunakan komunikasi data internet ataupun intranet (jaringan *LAN)*. Kemampuan untuk menyimpan data dan menampilkan data secara *visual* baik menggunakan angka, warna ataupun grafis semakin memudahkan dalam pembuatan *project* dibidang *Internet of Things*. Terdapat 3 komponen utama Blynk

a. Blynk Apps

Blynk Apps memungkinkan untuk membuat *project interface* dengan berbagai maca komponen *input output* yang mendukung untuk pengiriman maupun penerimaan data serta merepresentasikan data sesuai dengan komponen yang dipilih. *Representasi* data dapat berbentuk visual angka maupun grafik.

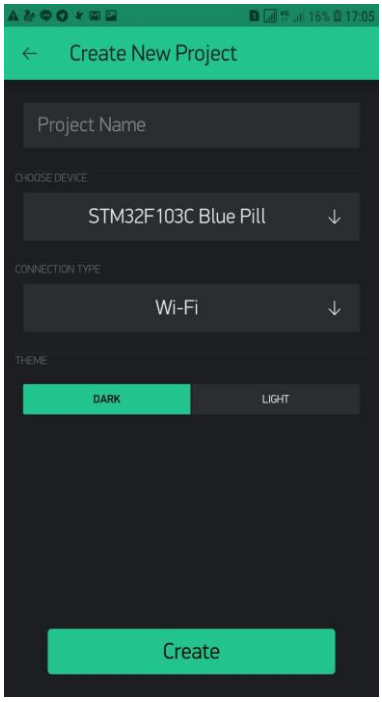
Terdapat 4 jenis *kategory* komponen yang berdapat pada Aplikasi Blynk

- *Controller* digunakan untuk mengirimkan data atau perintah ke *Hardware*
- *Display* digunakan untuk menampilkan data yang berasal dari *hardware* ke *smartphone*

• *Notification* digunakan untuk mengirim pesan dan notifikasi.

• *Interface* Pengaturan tampilan pada aplikasi Blynk dpat berupa menu ataupun tab.

• *Others* beberapa komponen yang tidak masuk dalam 3 kategori sebelumnya diantaranya *Bridge, RTC, Bluetooth*



Gambar 2.19. *Create New Project BLYNK*

b. *Blynk Server*

Blynk server merupakan fasilitas *Backend Service* berbasis *cloud* yang bertanggung jawab untuk mengatur komunikasi antara aplikasi *smart phone* dengan lingkungan *hardware*. Kemampun untuk menangani puluhan *hardware* pada saat yang bersamaan semakin memudahkan bagi para pengembang sistem IoT. *Blynk server* juga tersedia dalam bentuk *local server* apabila digunakan pada lingkungan tanpa internet.

c. *Blynk Library*

Blynk Library dapat digunakan untuk membantu pengembangan *code*. *Blynk library* tersedia pada banyak *platform* perangkat keras sehingga semakin memudahkan para pengembang IoT dengan *fleksibilitas hardware* yang didukung oleh lingkungan Blynk.

2.12. Perangkat Lunak

Arduino IDE *(Integrated Development Environment)* adalah *software* yang di gunakan untuk memprogram di arduino, dengan kata lain Arduino IDE sebagai media untuk memprogram *board* Arduino. Arduino IDE bisa di *download* secara gratis di *website* resmi Arduino IDE. Arduino IDE ini berguna sebagai *text editor* untuk membuat, meng*edit*, dan juga mevalidasi *kode* program. bisa juga digunakan untuk meng-*upload* ke *board* Arduino. *Kode* program yang digunakan pada Arduino disebut dengan istilah Arduino *"sketch"* atau disebut juga *source code* arduino, dengan ekstensi *file source code .ino*

STM32 merupakan *mikrokontroler* dari keluarga *STMicroelectronics*. Jadi semua metode yang ada untuk memprogram *chip* ARM dapat digunakan untuk papan STM32 juga. Salah satu IDE terkenal dan umum digunakan adalah Keil ARM MDK dan selain itu juga dapat menggunakan IAR, *Atollic TrueStudio, MicroC Pro ARM, ARM Crossworks, Ride 7, PlatformIO + STM32* dll. Namun yang membuat board ini sangat populer adalah kemampuannya untuk diprogram dengan Arduino IDE. Dengan cara ini orang dapat memulai dan membangun proyek dengan STM32 dalam waktu singkat karena banyak yang sudah terbiasa dengan Arduino IDE dan bahasa pemrograman yang mudah

digunakan dan banyak dukungan *library* yang tersedia. Untuk itu dalam tutorial ini akan menggunakan IDE Arduino untuk memulai dengan STM32.

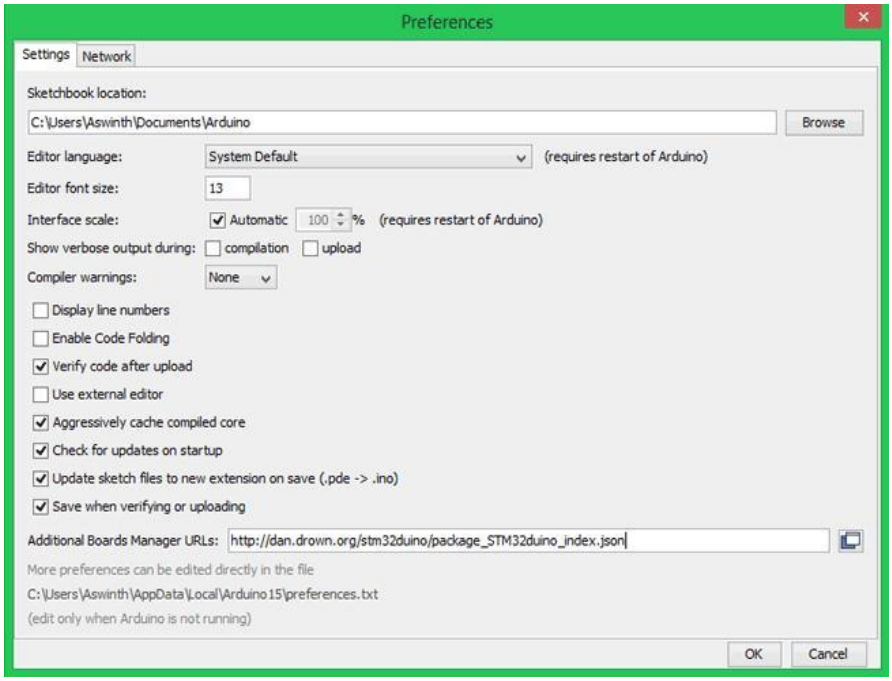
Ikuti langkah-langkah di bawah ini untuk mengunduh dan menyiapkan Arduino IDE untuk digunakan dengan papan Pengembangan STM 32.

Langkah 1: – Jika belum meng*instal* Arduino IDE, unduh dan instal dari tautan ini. Pastikan memilih sistem operasi yang benar.

Langkah 2: – Setelah Menginstal Arduino IDE buka dan unduh paket yang diperlukan untuk papan STM32. Ini dapat dilakukan dengan memilih File -> *Preferences*.

Langkah 3: – Mengklik Preferensi akan membuka kotak *dialog* yang ditunjukkan di bawah ini. Di kotak teks URL *Boards Manager* tambahan tempel tautan di bawah ini :

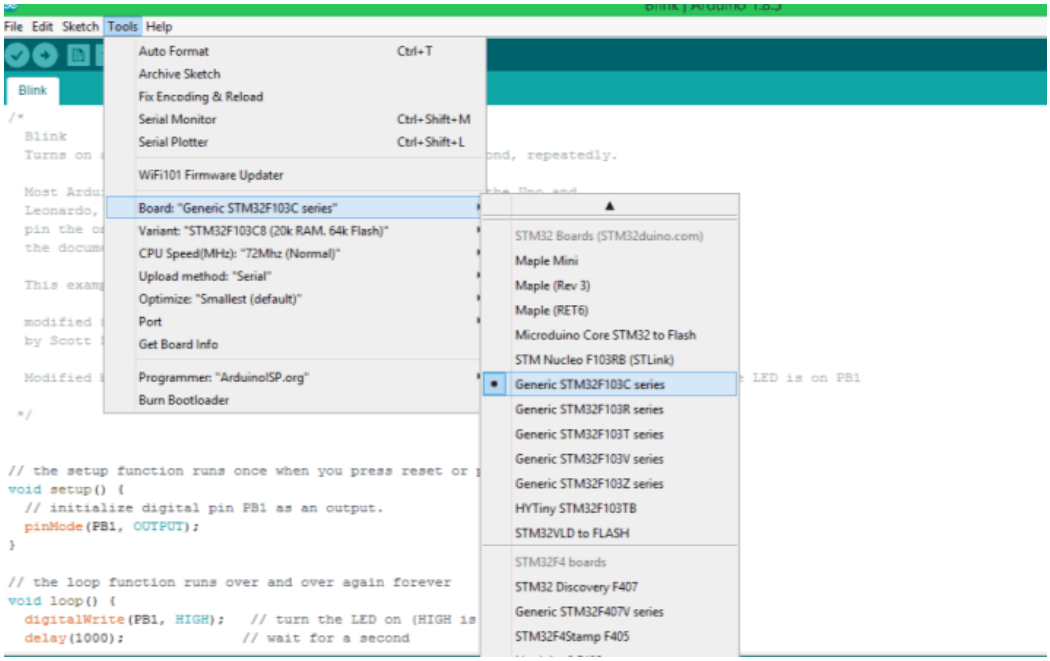
http://dan.drown.org/stm32duino/package_STM32duino_index.json



Gambar 2.20. *Preferences*.

Langkah 4: – Sekarang pergi ke *Tools -> Board -> Board Manajer*. Ini akan membuka kotak *dialog Boards manager*, mencari “STM32F1” dan meng*instal* paket yang muncul.

Langkah 5: – Setelah paket, *instalasi* selesai. Pergi ke *Tools* dan geser ke bawah untuk menemukan seri *Generic* STM32F103C seperti yang ditunjukkan di bawah ini. Kemudian pastikan *varian*nya adalah tipe 64k *Flash*, kecepatan CPU adalah 72MHz dan ubah metode *upload* ke *Serial.*



Gambar 2.21. Pilih *board STM32F103C series*.

Langkah 6: – Sekarang, hubungkan *board* FTDI ke komputer dan periksa *port* COM yang mana *board* FTDI terhubung menggunakan *device manager*. Kemudian, pilih nomor *port* yang sama di *Tools-> Port*

Langkah 7: – Setelah semua perubahan dilakukan, periksa sudut kanan bawah Arduino IDE dan Anda akan melihat pengaturan berikut sedang diatur. Papan FTDI terhubung ke *COM*7.

Sekarang Arduino IDE siap mem*program* STM 32 *Blue Pill Development Boards* [15].