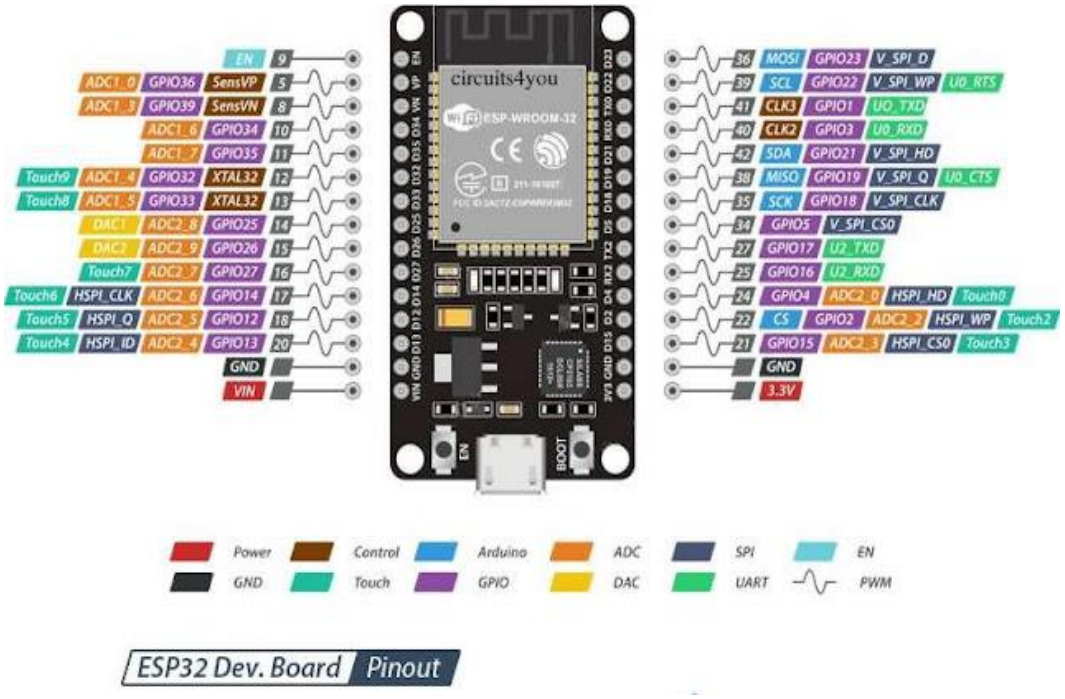
BAB II

LANDASAN TEORI

A. ESP32

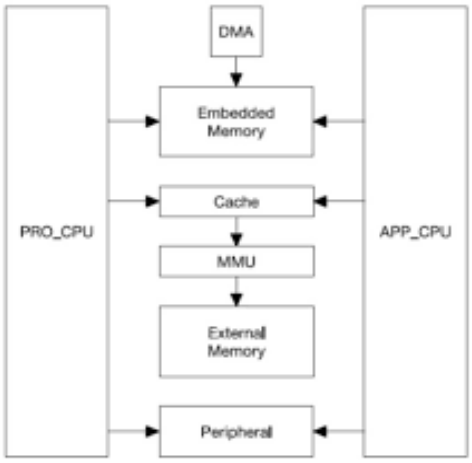
Espressif System memperkenalkan teknologi baru sebagai penerus ESP8266 adalah ESP32 dengan biaya rendah, daya system yang rendah pada chip mikrokontroler dengan terintegrasi *Wi-Fi*, kemampuan mode *Bluetooth* ganda dan lebih fleksibel dikarenakan hemat daya. ESP32 cocok digunakan untuk pengaplikasian *Internet Off Things* ternyata sebagai pilihan yang dapat diandalkan di lingkungan industri karena rentang suhu operasi yang luas. ESP32 dapat bertindak secara mandiri yang lengkap dan bias juga bertindak sebagai perangkat pendukung.



Gambar 2.1 ESP 32 [1].

Pada ESP 32 memiliki detail teknis dan fungsi seperti dijelaskan pada di bawah ini :

a) Sistem dan Memori ESP32 adalah sistem *dual-core* dengan dua CPU *Harvard Architecture Xtensa* LX6. Semua memori tertanam, memori eksternal dan *periferal* terletak di bus data dan / atau bus instruksi CPU ini.



Gambar 2.2 Sruktur sistem ESP32 [1].

Mikrokontroler memiliki dua inti - PRO_CPU untuk protokol dan APP_CPU untuk aplikasi. Ruang alamat untuk data dan bus instruksi adalah 4GB dan ruang alamat *periferal* adalah 512KB. Selain itu, memori yang disematkan adalah 448KB ROM, 520KB SRAM dan dua memori 8KB RTC. Memori eksternal mendukung hingga empat kali 16MB Flash.

b) Jam dan Timer ESP32 dapat menggunakan *Phase Lock Loop* (PLL) internal 320MHz atau kristal eksternal. Dimungkinkan juga untuk menggunakan sirkuit berosilasi sebagai sumber *clock* pada 2-40MHz untuk menghasilkan *clock* master CPU_CLK untuk kedua *core* CPU.

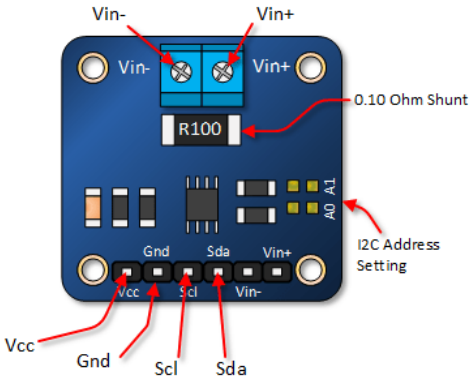
c) Diagram Blok dan Fungsi Struktur mikrokontroler ESP32 dirancang untuk beroperasi di bawah protokol berikut - TCP / IP, MAC WLAN 802.11 b/g/n/e/i WLAN penuh, dan spesifikasi Direct Wi-Fi. Mikrokontroler dapat menyediakan operasi *Basic Service Set* (BSS) STA dan *SoftAP* di bawah protokol Fungsi Kontrol Terdistribusi (DCF). Ini juga mendukung operasi grup P2P yang sesuai dengan protokol P2P Wi-Fi terbaru. Dengan demikian, bisa beroperasi sebagai stasiun dan terhubung ke internet atau *server* dan titik akses untuk menyediakan antarmuka pengguna untuk, misalnya, *smartphone* yang menjalankan aplikasi seluler.

d) Pemrograman ESP32. Sistem operasi waktu-nyata pada ESP32 adalah *FreeRTOS* yang merupakan *open source*, yang dirancang untuk sistem

tertanam dan menyediakan fungsi dasar untuk aplikasi tingkat yang lebih tinggi. Fungsi inti adalah manajemen memori, manajemen tugas dan sinkronisasi API [1].

B. Sensor INA219

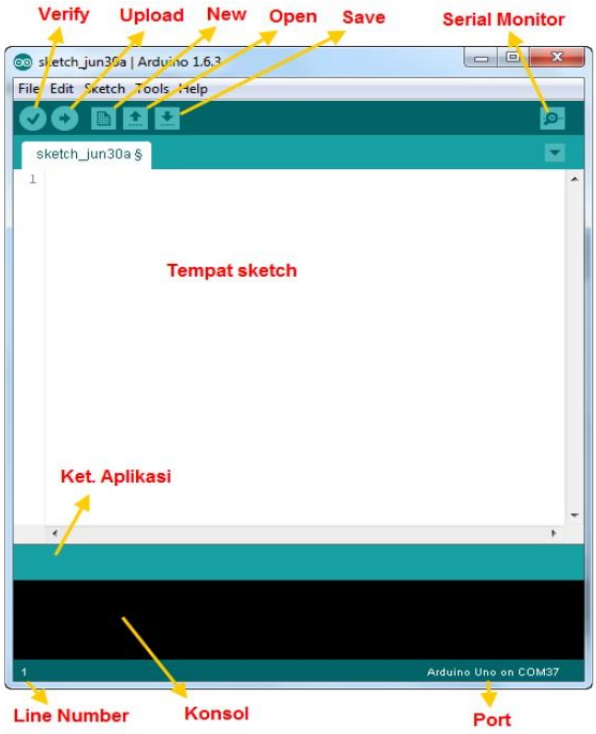
Sensor INA219 merupakan modul sensor yang mampu mengukur tegangan, arus dan daya secara bersamaan. Cara kerja sensor ini adalah arus yang dibaca mengalir melalui kabel tembaga yang terdapat didalamnya yang menghasilkan medan magnet yang di tangkap oleh IC medan terintregasi dan diubah menjadi tegangan proporsional. Karakterisasi sensor INA219 saat ini dilakukan menggunakan *Resistor* pasir dengan nilai 10W15RJ. Dimana hasil keluaran tegangan dan arus akan dilakukan pengulangan sebanyak tiga kali dan dibandingkan dengan nilai keluaran Avometer [2].



Gambar 2.3 INA219 [2].

C. *Arduino* IDE

Untuk memprogram *board* ESP32, kita butuh aplikasi IDE (*Integrated Development Environment*) bawaan dari *Arduino.* Aplikasi ini berguna untuk membuat, membuka, dan mengedit *source code* ESP32 (*Sketch*es, para programmer menyebut *source code Arduino* dengan istilah "*sketchs*"). *Sketch* merupakan *source code* yang berisi logika dan algoritma yang akan di*upload* ke dalam IC *mikrokontroller* [3].



Gambar 2.4 *Arduino* IDE [3].

Interface Arduino IDE tampak seperti gambar di atas. Dari kiri ke kanan dan atas ke bawah, bagian-bagian IDE *Arduino* terdiri dari:

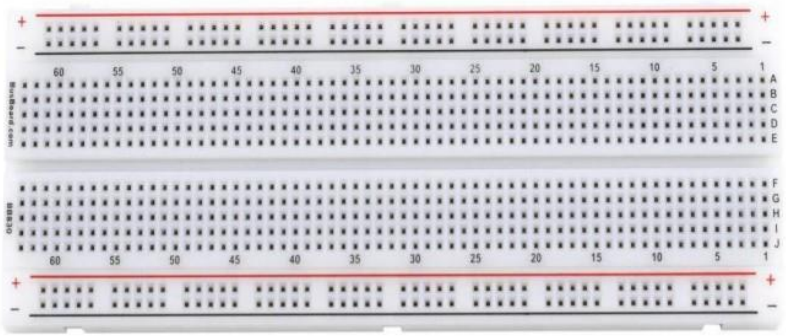
Verify : pada versi sebelumnya dikenal dengan istilah *Compile*. Sebelum aplikasi di*upload* ke *board Arduino*, biasakan untuk memverifikasi terlebih dahulu *sketch* yang dibuat. Jika ada kesalahan pada *sketch*, nanti akan muncul *error*. Proses *Verify / Compile* mengubah *sketch* ke *binary code* untuk di*upload* ke *mikrokontroller*.

- *Upload* : tombol ini berfungsi untuk meng*upload sketch* ke *board Arduino*. Walaupun kita tidak mengklik tombol *verify*, maka *sketch* akan di-*compile*, kemudian langsung di*upload* ke *board*. Berbeda dengan tombol *verify* yang hanya berfungsi untuk memverifikasi *source code* saja.
- *New Sketch* : Membuka window dan membuat *sketch* baru
- *Open Sketch* : Membuka *sketch* yang sudah pernah dibuat. *Sketch* yang dibuat dengan IDE *Arduino* akan disimpan dengan *ekstensi file .ino*
- *Save Sketch* : menyimpan *sketch*, tapi tidak disertai *mengcompile.*

- *Serial* Monitor : Membuka *interface* untuk komunikasi *Serial* , nanti akan kita diskusikan lebih lanjut pada bagian selanjutnya.
- Keterangan Aplikasi : pesan-pesan yang dilakukan aplikasi akan muncul di sini, misal "*Compiling*" dan "*Done Uploading*" ketika kita meng*compile* dan meng*upload sketch* ke board *Arduino*.
- Konsol : Pesan-pesan yang dikerjakan aplikasi dan pesan-pesan tentang *sketch* akan muncul pada bagian ini. Misal, ketika aplikasi meng*compile* atau ketika ada kesalahan pada *sketch* yang kita buat, maka informasi *error* dan baris akan diinformasikan di bagian ini.
- Baris *Sketch* : bagian ini akan menunjukkan posisi baris kursor yang sedang aktif pada *sketch*.
- Informasi *Port* : bagian ini menginformasikan *port* yang dipakah oleh *board Arduino* [3].

D. *Breadboard*

Breadboard adalah salah satu bagian paling mendasar ketika belajar bagaimana membangun sebuah rangkaian elektronik sederhana. *Breadboard* elektronik sebenarnya mengacu pada *Breadboard* tanpa solder. Ini adalah unit yang bagus untuk membuat rangkaian sementara dan pembuatan *prototipe*, dimana kita sama sekali tidak memerlukan solder. *Breadboard* dapat menampung sirkuit yang paling sederhana maupun yang sangat kompleks. Jika rangkaian melebihi papan, kita juga dapat dengan mudah mengganti dan menambahkan kompenen elektronika yang akan kita gunakan sehingga dapat menghemat waktu dan biaya [3].



Gambar 2.5 *Breadboard* [3].

E. *Jumper*

Jumper pada komputer adalah *connector* atau penghubung sirkuit *elektrik* yang digunakan untuk menghubungkan atau memutus hubungan pada suatu sirkuit. *Jumper* juga digunakan untuk melakukan *setting* pada papan *elektrik* [3].

Gambar 2.6 Kabel *Jumper* [3].

F.*Resistor*

Resistor merupakan komponen elektronika pasif yang berfungsi untuk membatasi jumlah aliran listrik yang mengalir dalam suatu rangkaian elektronika. Dalam penggunaannya *Resistor* dapat dirangkai menjadi suatu rangkaian yang dapat disusun secara seri atau paralel maupun keduanya. Rangkaian gabungan antara rangkaian seri *Resistor* dan juga rangkaian paralel *Resistor* disebut juga rangkaian kombinasi *Resistor* [4].

G. Baterai

Baterai adalah suatu proses kimia listrik, dimana pada saat pengisian energi listrik diubah menjadi kimia dan saat pengeluaran/*discharge* energi kimia diubah menjadi energi listrik[2]. Baterai menghasilkan listrik melalui proses kimia. Baterai atau *akkumulator* adalah sebuah sel listrik dimana didalamnya berlangsung proses elektrokimia yang *reversible* (dapat berkebalikan) dengan efisiensinya yang tinggi. Yang dimaksud dengan reaksi elektrokimia reversibel adalah didalam baterai dapat berlangsung proses pengubahan kimia menjadi tenaga listrik (proses pengosongan) dan sebaliknya dari tenaga listrik menjadi tenaga kimia (proses pengisian) dengan cara proses regenerasi dari elektroda-elektroda yang dipakai yaitu, dengan melewatkan arus listrik dalam arah polaritas yang berlawanan didalam sel. Baterai terdiri dari dua jenis yaitu, baterai primer dan baterai sekunder [5].