KOMPREHENSIF

PERANCANGAN TEMPAT PENITIPAN HELM OTOMATIS MENGGUNAKAN VOICE RECOGNITION BERBASIS ARDUINO

ACHMAD SYAIFUL KURNIADI

5215144150

PROGRAM STUDI PENDIDIKAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI JAKARTA

2017

ABSTRACT

ACHMAD SYAIFUL KURNIADI, Automatic Helmet Day Designing Using Arduino Based Voice Recognition. Comprehensive. Jakarta, Electronic Engineering Education Study Program, Department of Electrical Engineering, Faculty of Engineering, Jakarta State University, 2017. Supervisor, Muhammad Yusro, MT.

The authors designed an Automatic Helmet Care Planning Using Arduino Based Voice Recognition which aims to create a helmet day care automatically with the help of a conveyor device as a helmet carrier from the consumer to the storage box and as a helmet attachment from the storage box to the consumer.

In addition this tool uses voice commands as a command in running the system. Commands entered into EasyVR must be verified before they can be run. To be valid then previously EasyVR must be filled voice or direcord terlebuh first.

In addition to Arduino in this tool also serves very important in addition to the program output, Arduino also as a bridge between EasyVR with dc motor as well as between the sensor objects with dc motor.

Based on the design results that have been done, all the design of the tool has been completed and the tool is ready to be realized in the real application.

Keywords: Design, Custody, Voice Recognition, Arduino, Conveyor, DC Motor, EasyVR

ABSTRAK

ACHMAD SYAIFUL KURNIADI, Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*. Komprehensif. Jakarta, Program Studi Pendidikan Teknik Elektronika, Jurusan Teknik Elektro, Fakultas Teknik Universitas Negeri Jakarta, 2017. Dosen Pembimbing, Muhammad Yusro, MT.

Penulis merancang sebuah Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino* yang bertujuan untuk membuat suatu tempat penitipan helm secara otomatis dengan bantuan alat yaitu conveyor sebagai penghantar helm dari konsumen ke kotak penyimpanan dan sebagai penghatar helm dari kotak penyimpanan ke konsumen.

Selain itu alat ini menggunakan perintah suara sebagai perintah dalam menjalankan sistemnya. Perintah yang dimasukan kedalam *EasyVR* harus di verifikasi terlebih dahulu sebelum dapat dijalankan. Untuk dapat *valid* maka sebelumnya *EasyVR* harus diisikan suara atau direcord terlebuh dahulu.

Selain *Arduino* dalam alat ini juga berfungsi sangat penting selain untuk output program, *Arduino* juga sebagai penjembatan antara *EasyVR* dengan motor dc serta antara sensor benda dengan motor dc.

Berdasarkan hasil perancangan yang telah dilakukan, semua desain alat telah selesai dirancang dan alat siap untuk diwujudkan dalam aplikasi yang sesungguhnya.

Kata Kunci : Perancangan, Tempat Penitipan, *Voice Recognition*, *Arduino*, *Conveyor*, Motor DC, *EasyVR*.

LEMBAR PENGESAHAN

NAMA DOSEN	TANDA TANGAN	TANGGAL
Dr. Muhammad Yusro, MT (Dosen Pembimbing)	.....	13 Juli 2017

PENGESHAN PANITIA UJIAN KOMPREHENSIF

NAMA DOSEN	TANDA TANGAN	TANGGAL
Drs. Mufti Ma'sum, M.Pd (Ketua Sidang)	.....	13 Juli 2017
Drs. Jusuf Bintoro, MT (Sekretaris)	.....	13 Juli 2017
Aodah Diamah, M.Eng (Dosen Ahli)	.....	13 Juli 2017

Tanggal Lulus : 11 Juli 2017

LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa :

1. Karya tulis komprehensif saya yang berjudul "Perancangan Tempat Penitipan Helm otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*" ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik sarjana, baik di Universitas Negeri Jakarta maupun di perguruan tinggi lain.
2. Karya tulis ini adalah murni gagasan, rumusan, dan penelitian saya sendiri dengan arahan dosen pembimbing.
3. Karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai norma yang berlaku di Universitas Negeri Jakarta.

Jakarta, 19 Juli 2017

Yang Membuat Pernyataan

ACHMAD SYAIFUL KURNIADI

5215144150

KATA PENGANTAR

Dengan mengucapkan Alhamdulillahirabbil'alamiin serta puji dan syukur ke hadirat Allah SWT, karena dengan rahmat dan hidayahnya, sehingga saya dapat menyelesaikan makalah komprehensif yang berjudul "Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*" terlaksana dengan baik.

Penyusunan makalah ini dilakukan untuk memenuhi salah satu syarat untuk memperoleh gelar Sarjana Pendidikan Teknik Elektronika Fakultas Teknik Universitas Negeri Jakarta. Penulis menyadari bahwa penyusunan makalah ini masih belum sempurna sehingga penulis membutuhkan kritik dan saran untuk penyempurnaannya.

Penulis ingin menyampaikan ucapan terimakasih yang sebesar-besarnya kepada:

1. Bapak Drs. Pitoyo Yuliatmojo, MT, selaku Ketua Program Studi FT UNJ.
2. Bapak Dr. Muhammad Yusro, MT, selaku Dosen Pembimbing.
3. Kedua Orang Tua, Keluarga, Calon Istri serta Teman-teman ANTV yang telah memberikan semangat, motifasi, support serta do'a yang tidak pernah terhenti diucapkan untuk kelancaran dan keberhasilan.

Akhir kata, semoga Allah Subhanahu wa ta'ala membalas segala kebaikannya.

Penulis

Achmad Syaiful Kurniadi

5215144150

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

LAMPIRAN

Halaman

BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Pendidikan sangatlah penting apa lagi pendidikan di indonesia. Dengan pendidikan masyarakat indonesia lebih dapat mengembangkan kemampuan yang mereka punya. Teknologi saat ini berkembang sangat cepat sekali. Tidak heran jika penggunanya selalu merasa tertinggal. Karena begitu pesatnya, hingga kini mengikuti perkembangan teknologi sudah menjadi tuntutan logis agar bisa selalu *up to date* pada saat ini. Tidak hanya itu, teknologi itu pun memiliki tujuan yaitu untuk mempermudahkan manusia dalam hal pekerjaan.

Di Jakarta atau di perkotaan lain banyak sekali kejahatan, dari kejahatan perampasan kendaraan bermotor hingga kejahatan pencurian helm. Menurut sumber berita online, Setahun meresahkan, mahasiswa pencuri helm Ini ditangkap polisi. Polisi menggelar rilis kasus pencurian helm di Mapolrestabes Semarang.

Menurut sumber berita online yang lain polsek tasikmadu memberikan sosialisasi kepada petugas parkir agar lebih waspada terutama di pusat kegiatan dan keramaian masyarakat seperti di kolam renang Intanpari, sebagai antisipasi terhadap pencurian helm dan motor.

Dari informasi berita online tersebut banyak memberikan informasi tentang penangkapan pencurian helm serta himbauan untuk penjaga parkir. Untuk mencegah terjadinya pencurian helm maka disetiap tempat penitipan kendaraan bermotor ada tempat penitipan helm sebagai salah satu solusi untuk mencegah kejahatan. Oleh karena itu saya membuat sebuah trobosan baru yaitu perancangan

tempat penitipan helm otomatis menggunakan *Voice Recognition* berbasis *Arduino*. Di mana alat ini belum pernah dijumpai di tempat penitipan helm. Kenapa saya menggunakan *Voice* dikarenakan setelah saya meriset beberapa paper, saya tertarik dengan aplikasi yang satu ini yaitu aplikasi *EasyVR.*

Di mana aplikasi ini masih jarang dipergunakan, aplikasi ini baru dipergunakan pada *voice controlled smart home system* (Amrutha et al., 2015) dari india yang mengggunakan alat ini untuk membantu mempermudah orang tua dan penyandang cacat, dengan sistem yang dapat merespon perintah suara dan mengontrol *on/off* status perangkat listrik, seperti lampu, kipas angin, televisi dll di rumah. Selanjutnya *Arduino based voice controlled* robot (Kannan, 2015) dari india yang menggunakan alat ini pada sebuah robot yang pergerakannya dapat dikontrol oleh pengguna dengan memberikan perintah suara tertentu. Robot ini dipekerjakan pada semua jenis lingkungan yang tercemar, kimia serta nuklir.

Yang terakhir keamanan pintu pagar otomatis menggunakan *Voice Recognition* (N, Ponco, & Sc, 2014.) dari Indonesia yang menggunakan alat ini untuk memudahkan dalam membuka dan menutup pintu tanpa harus mendorong secara manual ketika sedang membawa belanjaan yang banyak. Dimana kesimpulan aplikasi *EasyVR* yang mereka gunakan untuk mengontrol segala macam peralatan dari tv, lampu, kipas angin, pintu garasi hingga pergerakan sebuah robot.

1.2 Identifikasi Masalah

1. Adanya konsumen atau pelanggan yang kecewa terhadap keamanan pada helmnya.
2. Adanya konsumen atau pelangan yang menyesalkan jika helmnya tergores.
3. Sistem yang digunakan dalam meletakan dan mengambil helm saat ini masi menggunakan sistem manual sepenuhnya.

1.3 Pembatasan Masalah

1. Perancangan dan pembuatan sistem kontrol menggunakan *Arduino Uno*, *Arduino Mega* serta *EasyVR*.
2. Modul *Voice Recognition* yang digunakan adalah tipe *EasyVR*3.
3. Modul sensor yang digunakan adalah Photodiode dan Led.
4. Suara yang digunakan hanya dapat 1 orang saja.

1.4 Perumusan Masalah

Bagaimana membuat rancangan tempat penitipan helm otomatis menggunakan *Voice Recognition* berbasis *Arduino*.

1.5 Tujuan Penulisan

1. Membuat rancangan tempat penitipan helm otomatis.
2. Menggunakan sistem mikrokontroler *Arduino* sebagai *converter Voice Recognition*.
3. Mengamankan helm dari tindak kejahatan.

1.6 Manfaat Penulisan

1. Mempermudah dalam pengambilan dan penitipan helm.
2. Tingkat keamanan untuk helm semakin tinggi.
3. Sebagai pengembangan teknologi terbaru berdasarkan teknologi yang sudah pernah ada.

BAB II

TINJAUAN PUSTAKA

2.1 *Conveyor*

Conveyor adalah alat pemindah barang yang paling efisien dibanding truck, tergantung design *conveyor*nya. Material yang dikirim dapat berupa apa saja tergantung meterial dan kekuatan *belt* pada *conveyor*nya. Keuntungan menggunakan *conveyor* adalah mudah dalam pengoperasiannya dan perawatannya, tetapi belt pada *conveyor* tidak dapat menahan suhu di atas 200^0C. Dengan *conveyor* material dapat dipindah-pindahkan kearah mana saja ia mau meletakan atau dimana saja ia bisa mengambilnya. Contoh jenis *belt* pada *conveyor* adalah *metal belt*, steel *cord belt*, dan *rubber belt*. *Belt* yang serimg di gunakan adalah berjenis *belt textil rubber belt*. Dimana prinsip kerja dari *belt* adalah mengirimkan material yang ada di atas *belt*, belt di gerakan oleh driver dengan menggunakan motor penggerak. Tampilan *conveyor* dapat dilihat pada gambar 2.1.

Gambar 2.1 Tampilan *Conveyor*

(Sumber : http://baker-online.com)

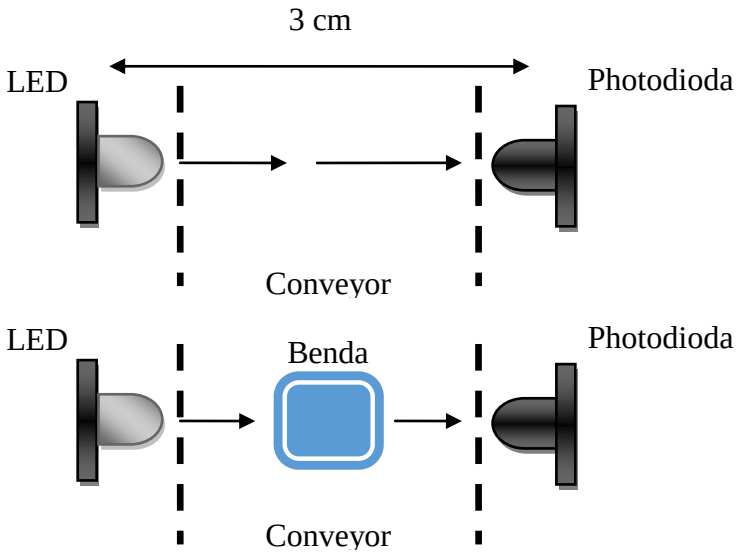
2.2 Sensor Benda

Sensor yaitu sesuatu alat yang digunakan untuk mendeteksi adanya perubahan pada lingkungan. *Tranducer* berasal dari kata "*tranducere*" artinya dalam bahasa latin yaitu mengubah, sehingga *tranducer* dapat didefinisikan sebagai suatu alat yang dapat mengubah suatu energi ke bentuk energi lainnya.

Didalam sensor benda terdapat Photodioda dimana Photodioda yaitu komponen elektronika yang dapat mengubah cahaya menjadi arus listrik. Photodioda merupakan komponen aktif yang terbuat dari bahan semikonduktor dan masih dalam keluarga dioda. Photodioda mempunyai dua kaki yang pertama kaki katoda dan kaki anoda, selain itu Photodioda mempunyai lensa dan filter optik yang terpasang dipermukaan sebagai pendeteksi cahaya.

Photodioda dapat mendeteksi cahaya yang berupa sinar inframerah, sinar ultra-violet, sinar x dan cahaya matahari. Oleh karena itu, Photodioda yang dapat mendeteksi berbagai cahaya ini telah banyak diaplikasikan ke berbagai perangkat Elektronika dan listrik seperti penghitung kendaraan, sensor cahaya kamera, alat-alat medis, *scanner barcode* dan peralatan keamanan. Motor penarik helm membutuhkan sensor benda untuk dapat mengaktifkan motor sehingga helm dapat masuk atau tertarik kedalam kotak penyimpanan. Oleh karenanya digunakanlah sebuah sensor benda. Sensor benda adalah sensor yang dapat aktif apabila sensor tersebut mendeteksi sebuah benda. Sensor benda dapat aktif ketika mendeteksi sebuah benda dengan jarak 3cm atau lebih. Kita dapat memanfaatkannya dalam membuat sensor benda dengan meletakan berupa LED *superbright* sebagai

pemancar dan Photodioda sebagai penerima. Tampilan sensor benda dapat dilihat pada gambar 2.2.



Gambar 2.2 Tampilan Sensor Benda

Pada gambar 2.2 LED dipasang berseberangan dengan Photodioda pada jarak sekitar 3cm. Apabila Photodioda terkena cahaya yang berasal dari pancaran LED maka hambatan Photodioda akan mengecil. Apabila cahaya LED yg dipancarkan ke Photodioda terpotong atau terhalang oleh sebuah benda, akibatnya Photodioda tidak menerima cahaya sehingga hambatan Photodioda bernilai tinggi.

2.3 Arduino

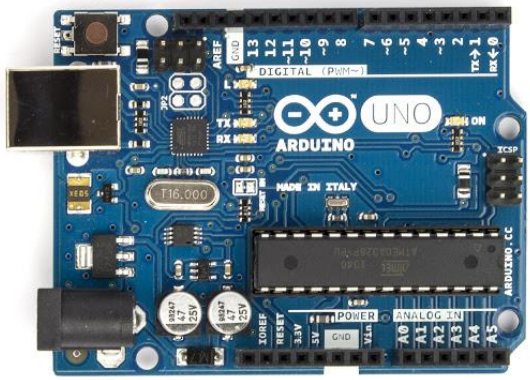
Arduino adalah *board* elektronik *open source* yang dirancang khusus untuk mempermudah bagi siapapun yang tertarik dalam menciptakan objek atau menggembangkan perangkat elektronik dimana elektronik tersebut dapat berinteraksi dengan berbagai macam sensor dan pengendali. *Arduino* juga

mempunyai komponen utama yaitu sebuah chip mikrokontroler dengan jenis AVR dari perusahaan ATmel.

Dimana mikrokontrolet itu sendiri adalah IC yang bisa diprogram dengan menggunakan komputer. Mikrokontroler juga suatu alat elektronika digital yang mempunyai masukan serta keluaran dan dikendalikan dengan program yang bisa ditulis dan dihapus dengan cara khusus. Mikrokontroler merupakan komputer yang berada didalam chip yang digunakan untuk mengontrol peralatan elektronik, dimana menekankan efektifitas biaya dan efisiensi. Mikrokontroler juga bisa disebut dengan kata lain "pengendali kecil" dimana sebuah sistem elektronik yang sebelumnya memerlukan banyak komponen-komponen seperti IC TTL dan CMOS dapat diperkecil dan terpusat serta dapat dikendalikan oleh mikrokontroler. Jadi mikrokontroler berfungsi sebagai "otak" yang mengendalikan input, proses dan output pada rangkaian elektronik.

2.3.1 *Arduino Uno*

Arduino Uno merupakan sebuah board mikrokontroler yang dikendalikan oleh ATmega328. *Arduino Uno* mempunyai 14 pin digital input/output, 6 pin digunakan untuk output PWM, 6 input analog, satu buah *oscilator* kristal 16 MHz, satu buah *port* USB, satu buah ICSP *header*, dan satu buah tombol *reset*. *Arduino Uno* sangat mudah dihubungkan ke komputer dengan menggunakan sebuah USB dan dapat mensuplainya dengan sebuah adaptor AC to DC. Tampilan gambar *Arduino Uno* dapat dilihat pada gambar 2.3.



Gambar 2.3 Tampilan *Arduino Uno*

(Sumber : https://arduino.cc)

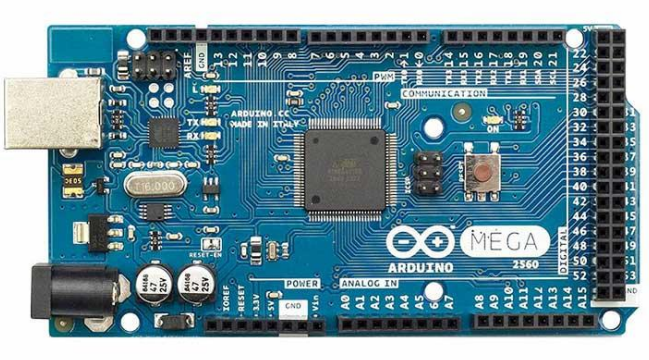
Uno dalam bahasa italianya adalah satu, satu itu sebagai penanda keluaran produk *Arduino*.

2.3.2 *Arduino ATMega* 2560

Arduino Mega 2560 adalah *board Arduino* yang merupakan perbaikan dari *board Arduino Mega* sebelumnya. Dimana *Arduino Mega* awalnya menggunakan chip ATmega 1280 kemudian diganti dengan chip ATmega 2560, oleh sebab itu namanya diganti menjadi *Arduino Mega* 2560. Dilihat dari fisik, ukuran *Arduino Mega* 2560 kurang lebih dua kali dari besar *Arduino Uno*, dan mempunyai pin digital dan analog yang lebih banyak di banding *Arduino Uno*.

Arduino Mega 2560 mempunyai pin I/O sebanyak 54 buah dimana 15 pin PWM, 16 pin input analog, 4 pin UART. *Arduino Mega* 2560 juga dilengkapi dengan satu buah *oscilator* 16 MHz, satu buah *port* USB, satu buah ICSP dan satu buah tombol *reset*. *Arduino* mempunyai beberapa tipe yaitu *Arduino Duemilanove*, *Arduino Uno*, *Arduino Leonardo*, *Arduino Mega* 2560, *Nano* R3,

Mini ATmega, *Pro Micro A*T dan *Esplora*. Tampilan gambar *Arduino Mega* 2560 dapat dilihat pada gambar 2.4.



Gambar 2.4 Tampilan *Arduino Mega* 2560

(Sumber : https:// arduino.cc)

2.3.3 Karakteristik *Arduino Uno*

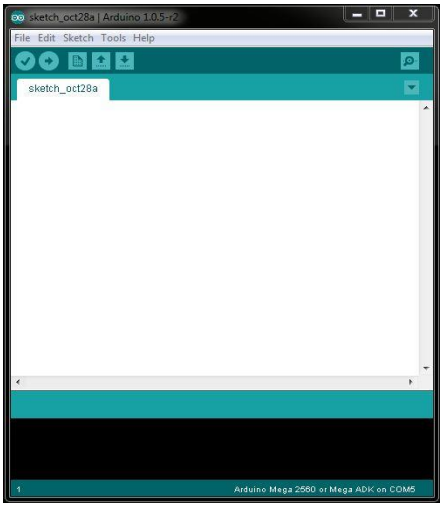
PCB *Arduino Uno* mempunyai panjang sekitar 2.7 dan 2.1 inci, mempunyai konektor USB. Mikrokontroler ATmega 328 dengan daya 5V. Pin digital I/O sejumlah 14, dimana 6 pin output PWM dengan 8 bit, 6 pin input analog. Serial 0 RX dan 1 TX. Digunakan untuk menerima RX dan untuk mengirim TX.

2.3.4 Karakteristik *Arduino Mega 5260*

Arduino Mega 2560 mempunyai bentuk yang lebih lebar yaitu 4x2,1 inci dimana dilengkapi dengan konektor USB. Arduino Mega 2560 memiliki 4 *port* serial. *Arduino Mega* 2560 dapat beroperasi dengan tegangan 5V, jika tegangan kepada *Arduino Mega* melebihi batas 12V maka akan merusak IC pada *Arduino*.

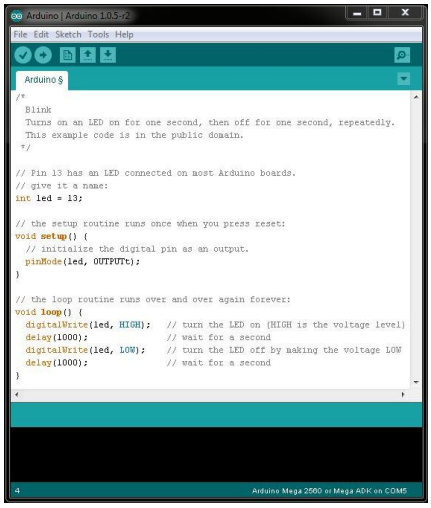
2.3.5 Cara Pemograman *Arduino*

1. Buka *Arduino* pada laptop atau komputer setelah itu akan muncul gambar seperti di bawah ini. Pada layar *skecth* anda bisa langsung mengetik program yang anda inginkan. Tampilan awal program *Arduino* dapat dilihat pada gambar 2.5.



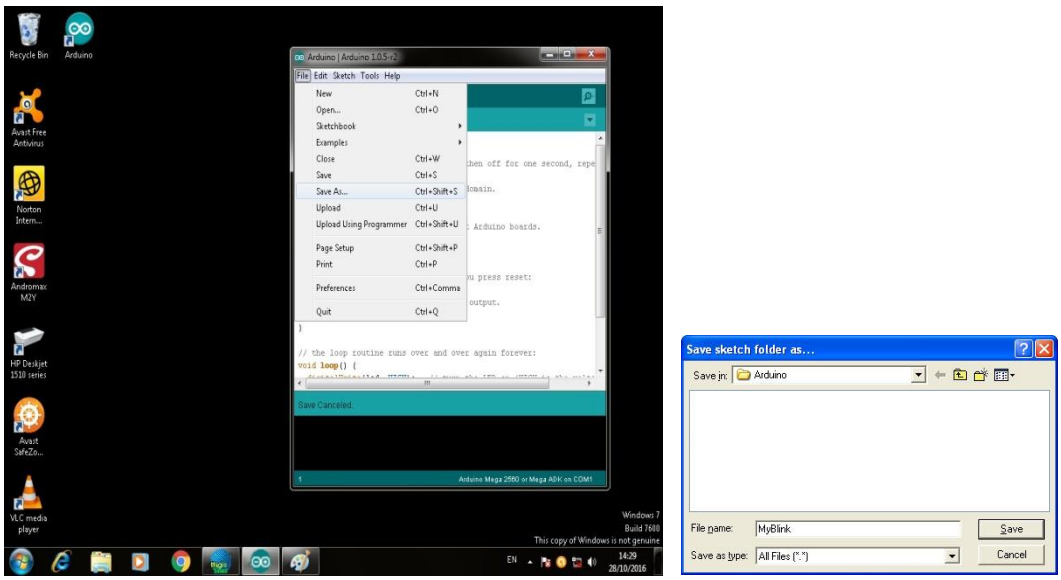
Gambar 2.5 Tampilan Awal Program *Arduino*

2. Silahkan coba ketik program sederhana berikut ini. Tampilan contoh program *Arduino* dapat dilihat pada gambar 2.6.



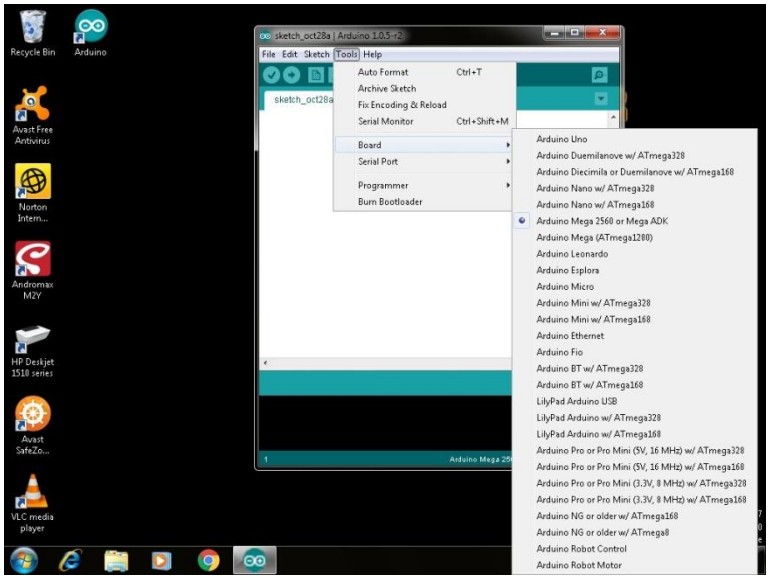
Gambar 2.6 Tampilan Contoh Program *Arduino*

3. Simpan program dengan menekan tombol *save* seperti biasa. Tampilan cara menyimpan program dapat dilihat pada gambar 2.7.



Gambar 2.7 Tampilan Menyimpan Program

4. Sesuaikan tipe *board* dengan sistem minimum yang digunakan. Klik *tools board Arduino*. Tampilan dalam memilih tipe *Arduino* dapat dilihat pada gambar 2.8.

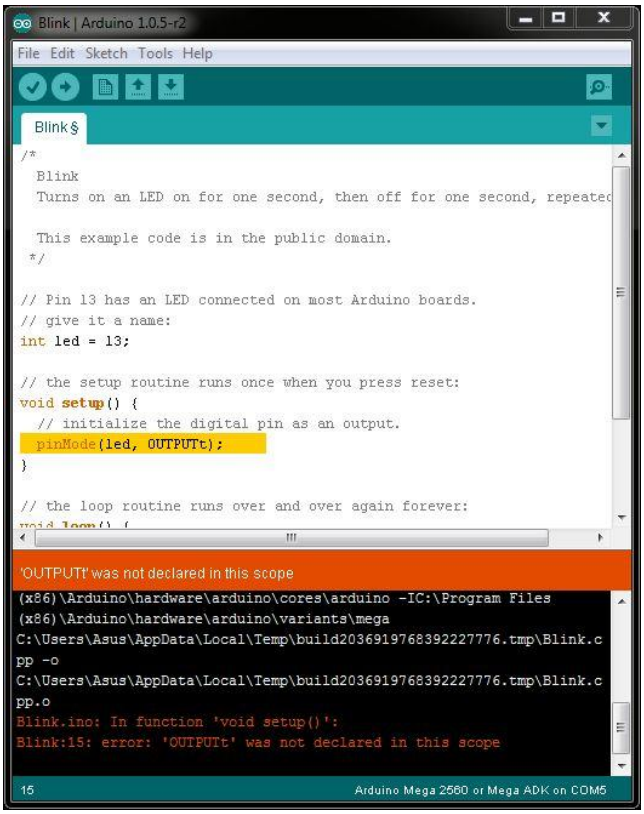


Gambar 2.8 Tampilan Memilih Tipe *Arduino*

5. Lakukan *verify* program untuk menjalankannya dan mengetahui apakah ada kesalahan pada program yang kita buat. Tampilan kompilasi program dapat dilihat pada gambar 2.9.

Gambar 2.9 Tampilan Kompilasi Program

6. Jika ada kesalahan pada penulisan program maka akan keluar tulisan berwarna merah pada bagian bawah *Arduino*. Tampilan kesalahan dalam program *Arduino* dapat dilihat pada gambar 2.10.



Gambar 2.10 Tampilan Kesalahan Program Pada *Arduino*

2.4 *EasyVR*

EasyVR adalah sebuah modul *Voice Recognition* dengan fungsi yang banyak. Dapat digunakan pada banyak aplikasi pengontrolan yang membutuhkan pendeteksian bukan hanya suara melainkan percakapan. *EasyVR* adalah generasi penerus setelah generasi pertama yang ada dipasaran yaitu *VRBot*. Modul ini dapat dihubungkan dengan board mikrokontroler *Arduino*. Dapat digunakan untuk berbagai macam aplikasi, contohnya mengontrol nyala lampu, kunci pintu, dan menyalakan serta mematikan televisi. Bisa juga sebagai modul pelengkap sensor pendengaran robot. *EasyVR* didukung berbagai bahasa yaitu: *English*, *Italian*, *German*, *French*, *Spanish*, dan *Japanese*.

Tampilan gambar *EasyVR* dapat dilihat pada gambar 2.11.

Gambar 2.11 Tampilan EasyVR

(Sumber : EasyVR, 2017)

EasyVR dapat digunakan atau dihubungkan dengan board mikrokontroler *Arduino*. *EasyVR* juga sangat cocok dipakai untuk beragam aplikasi, contohnya

untuk menyalakan lampu, mengunci pintu, menyalakan dan mematikan televisi. Parameter pada *EasyVR* dapat dilihat pada tabel 2.1

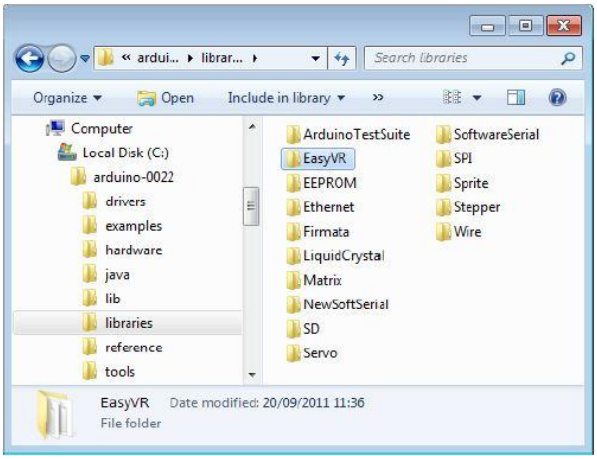
Tabel 2.1 Parameter *EasyVR*

Voltage	4.5 - 5.5V
Current	< 40mA.
Digital Interface	5V TTL level UART interface
Analog Interface	3.5mm mono-channel microphone connector + microphone pin interface
Size	30mm x 47.5mm
Recognition Accuracy	99% (under ideal environment)

(Sumber : EasyVR, 2017)

2.4.1 Cara sebelum menggunakan modul *EasyVR* ke *Arduino*

1. Hubungkan modul *EasyVR* ke *Arduino* seperti yang telah diuraikan sebelumnya.
2. Hubungkan *microphone* yang telah disertakan dengan modul *EasyVR* ke konektor MIC (J3).
3. Salin (*copy*) *library* modul *EasyVR* ke *folde*r *library Arduino* yang ada di komputer.
4. Kemudian hubungkan *Arduino* ke komputer melalui kabel *USB*. Tampilan Memindahkan *Library EasyVR* Ke *Library Arduino* dapat dilihat pada gambar 2.12.

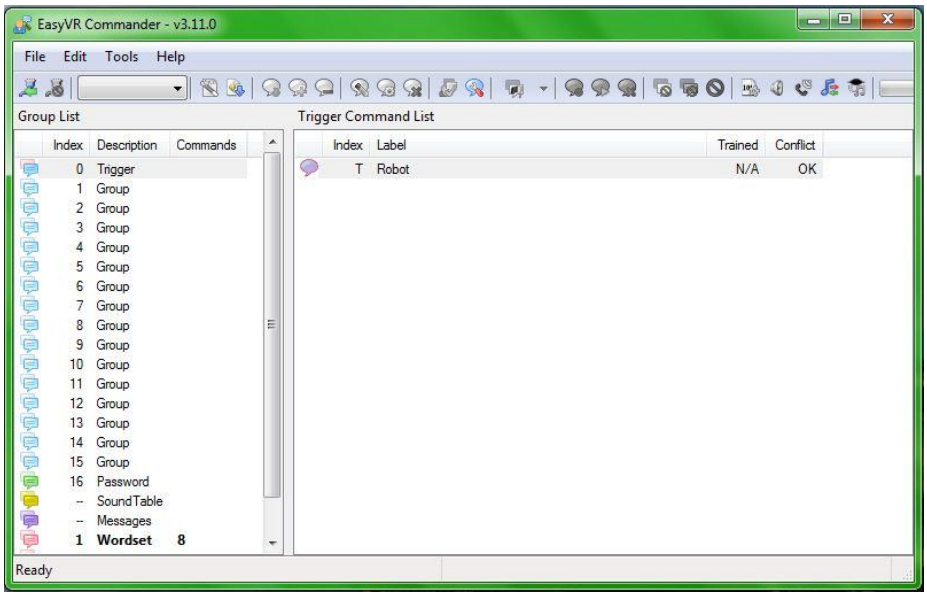


Gambar 2.12 Tampilan Memindahkan *Library EasyVR* Ke *Library Arduino*

5. Instal *Easy Commander* 3.4.10 dengan mengikuti petunjuk penginstalan sampai dengan selesai. Tampilan jendela instalasi program *EasyVR Commander* dapat dilihat pada gambar 2.13.

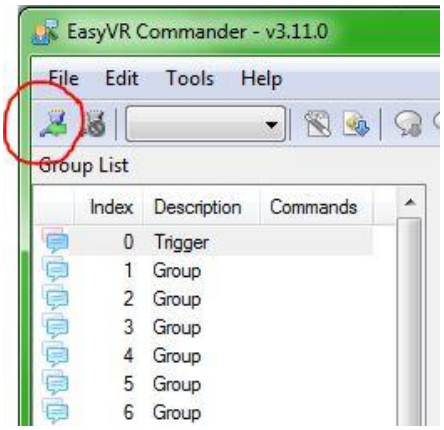
Gambar 2.13 Tampilan Jendela Instalasi Program *EasyVR Commander*

6. Buka program *EasyVR Commander* 4.3.10 yang telah terinstal dengan benar. Tampilan jendela *EasyVR Commander* dapat dilihat pada gambar 2.14.



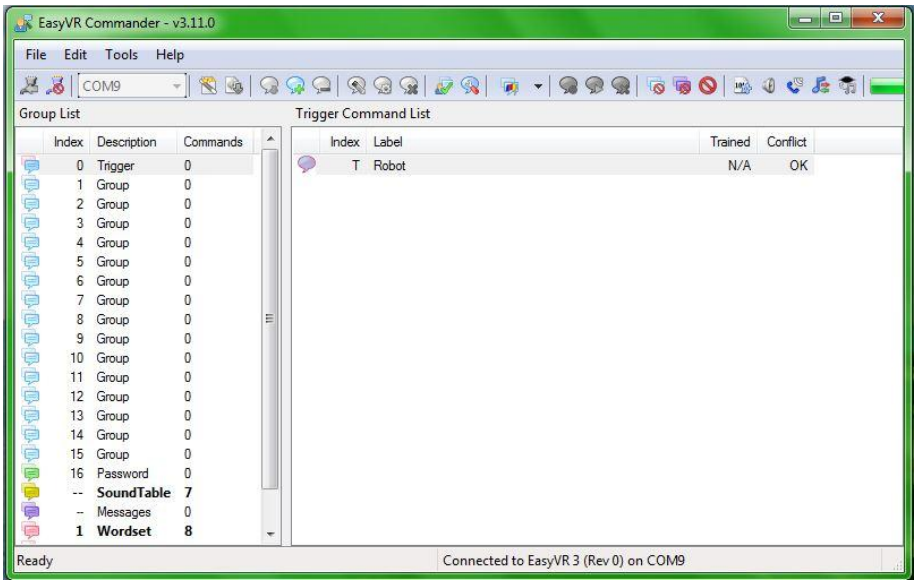
Gambar 2.14 Tampilan Jendela *EasyVR Commander*

7. Setelah *Arduino* tersambung dengan modul *EasyVR*, Klik *icon tool connect.* Tampilan tombol *Connect* dapat dilihat pada gambar 2.15.



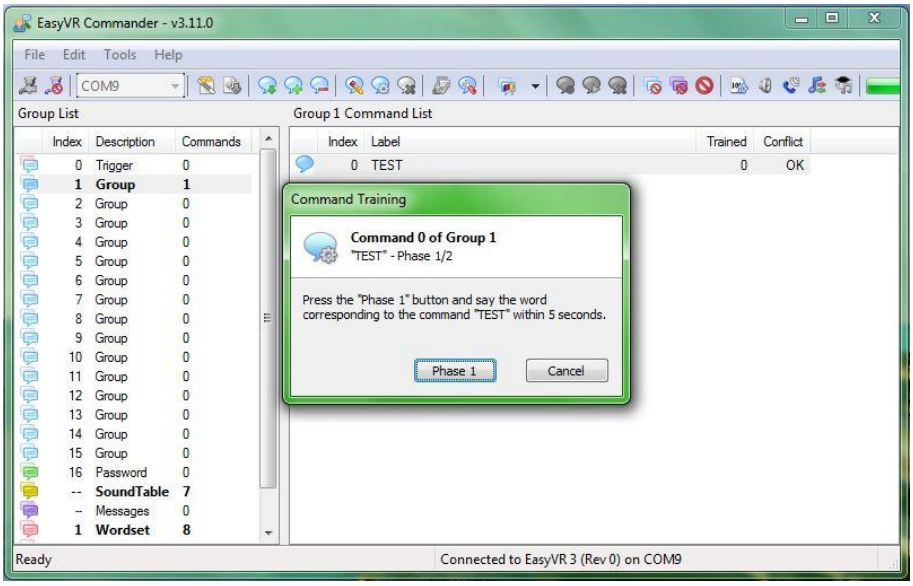
Gambar 2.15 Tampilan Tombol *Connect*

8. Setelah diklik tombol *connect* maka seharusnya akan terdapat tulisan *connected to EasyVR on COM9*, Tampilan modul *EasyVR* telah tersambung dengan *EasyVR Commander* dapat dilihat pada gambar 2.16.



Gambar 2.16 Tampilan Modul *EasyVR* Telah Tersambung Dengan *EasyVR Commander*

9. Setelah modul *EasyVR* tersambung dengan *EasyVR Commander*, langkah berikutnya adalah membuat *sample* suara yang akan digunakan sebagi perintah. Perekaman suara dapat dilakukan dengan cara merekam suara pada *EasyVR Commander*. Klik *icon tool "add command"*, kemudian buat label sesuai dengan keinginan. Klik *icon tool "train command"*, dan lakukan perekaman melalui *microphone* yang terdapat pada modul *EasyVR* dengan mengikuti instruksi yang terdapat pada jendela tampilan *EasyVR Commander*, lakukan perekaman suara sesuai dengan label yang telah diisi dan lakukan perekaman hingga dua kali perekaman. Tampilan instruksi perekaman pada *EasyVR Commander* dapat dilihat pada gambar 2.17.

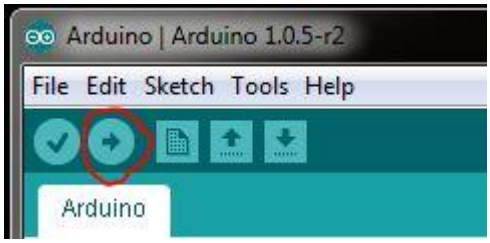


Gambar 2.17 Tampilan Instruksi Perekaman Pada *EasyVR Commander*

10. Setelah melakukan perekaman suara dengan baik, maka langkah selanjutnya adalah *men-Generate* kode pada *icon tool* di *EasyVR Commander*, kemudian simpan hasil dari *Generate* kode tersebut pada *folder* yang kita inginkan.
11. Hasil dari *Generate* kode adalah berbentuk bahasa *Arduino IDE*, berekstensi .pde. sehingga untuk mengedit dan meng-*upload* ke *Arduino* kita harus membuka *Arduino IDE*.
12. Setelah proses editing selesai, misalnya dengan penambahan fungsi *LCD* dan fungsi lain yang dikehendaki, untuk mengecek apakah proses editing benar, kita perlu mengklik *icon tool verify* pada *Arduino IDE*. Tampilan tombol *verify* pada *Arduino* dapat dilihat pada gambar 2.18.

Gambar 2.18 Tampilan Tombol *Verify* Pada *Arduino IDE*

13. Setelah proses *verify* selesai dilakukan setelah itu langkah selanjutnya adalah meng uploade program tersebut pada *Arduino* dengan mengklik *icon tool upload* pada *Arduino IDE*. Tampilan tombol Upload pada *Arduino* dapat dilihat pada gambar 2.19.



Gambar 2.19 Tampilan Tombol *Upload* Pada *Arduino IDE*

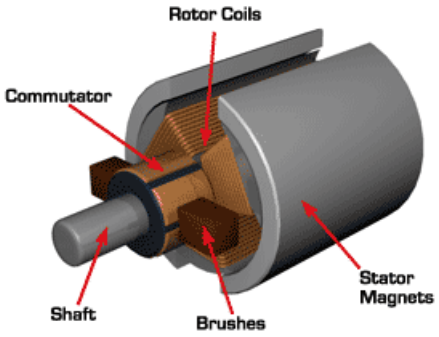
2.5 Motor DC

Motor DC adalah sebuah mesin listrik yang berfungsi mengubah tenaga listrik DC menjadi tenaga mekanik atau gerak. Tenaga gerak tersebut beruba motor. Motor DC ada dua bagian yang pertama ada bagian yang namanya stator dan yang kedua namanya rotor. Pengertian rotor adalah bagian yang berputar yang berupa koil dimana arus listrik dapat mengalir. Pengertian stator adalah bagian motor DC yang tetap dan menghasilkan sebuah medan magnet dari koilnya. Prinsip dasar dari motor DC adalah jika sebuah kawat berarus diletakan antara

kutub magnet utara dan selatan, maka pada kawat itu akan bekerja suatu gaya yang akan menggerakan kawat itu.

Arah putaran motor DC adalah apabila gerak kawat dapat ditentukan dengan “kaidah tangan kiri” yang berbunyi “apabila tangan kiri terbuka dan diletakan di antara kutub utara dan kutub selatan sehingga garis-garis gaya yang keluar dari kutub magnet menembus telapak tangan kiri dan arus didalam kawat mengalir searah dengan keempat jari, maka kawat akan mendapat gaya yang arah nya sesuai dengan arah ibu jari”. Gaya ini disebut gaya Lorentz.

Jika untuk membalik arah putaran pada motor DC adalah dapat dilakukan dengan membalik polaritas sumber tegangan inputnya. Dengan cara tersebut maka arus akan terbalik sehingga arah putaran motor juga akan terbalik. Tampilan bagian motor DC dapat dilihat pada gambar 2.20.



Gambar 2.20 Tampilan Bagian-bagian Motor DC

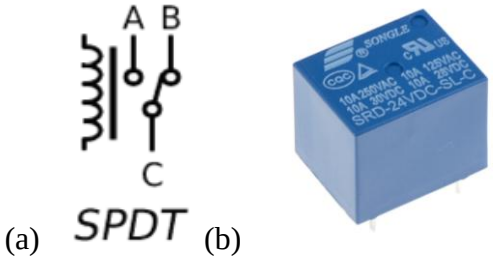
(Sumber : EasyVR, 2017)

2.6 Relay SPDT

SPDT adalah Single Pole Double Throw adalah saklar yang pengoperasiannya dilakukan oleh listrik dan relay ini terdiri dari dua yang

pertaman adalah koil dan yang kedua adalah mekanik. Prinsip kerja relay adalah ketika elektromagnetik menggerakan kontak saklar relay walau dengan arus listrik yang kecil relay akan tetap dapat menghantarkan listrik yang bertegangan lebih besar.

Relay mempunyai lima terminal, yaitu dua buah kumparan atau koil atau tiga terminal saklar yang dapat terhubung dan terputus dengan satu terminal pusat. Tampilan simbol dan bentuk fisik relay SPDT dapat dilihat pada gambar 2.21.



(a) (b)

Gambar 2.21 Tampilan (a) Simbol dan (b) Bentuk fisik SPDT

(Sumber : http://teknikelektronika.com)

2.7 *Power Supply*

Power Supply atau disebut catu daya adalah suatu alat listrik yang dapat menyediakan energi listrik untuk perangkat elektronika. *Power Supply* dibagi menjadi dua yaitu *power supply linier* dan *power supply switching*. Dimana *power supply linier* merupakan *power supply* yang sering digunakan. Cara kerja *power supply* ini adalah mengubah suatu tegangan AC yang bernilai besar menjadi tegangan AC yang bernilai kecil dengan bantuan transformator. Lalu tegangan ini disearahkan dengan menggunakan rangkaian penyearah tegangan tidak lupa dibagian akhirnya diberikan kapasitor untuk membantu menstabilkan tegangan, sehingga tegangan DC yang dihasilkan oleh *power supply* ini tidak terlalu

bergelombang. Untuk rangkaian *power supply* yang lainnya menggunakan sebuah regulator tegangan sehingga tegangan yang dihasilkan oleh *power supply* jauh lebih baik dari pada rangkaian yang hanya menggunakan dioda saja. *Power supply* ini dapat menghasilkan tegangan antara 0V sampai dengan 30V dengan arus antara 0A hingga 5A.

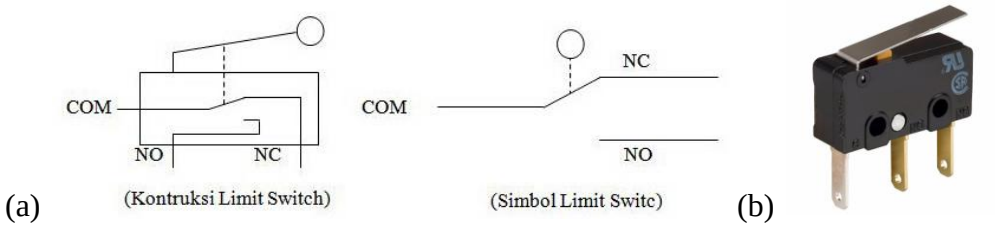
Jika *power supply* jenis switching ini menggunakan metode yang beda dengan *power supply linier*. Pada *power supply switching* tegangan AC yang masuk ke dalam rangkaian langsung disearahkan oleh rangkaian penyearah tanpa menggunakan bantuan dari transformer. Dimana cara menyearahkanya ialah dengan menggunakan frekuensi yang tinggi dari 10KHz sampai 1MHz. Selain itu pada rangkaian *power suply switching* di berikan sebuah *feedback* agar tegangan dan arus yang keluar dari rangkaian ini dapat dikontrol dengan baik. Selainitu ukuran *power supply* ini jauh lebih kecil dibandingkan dengan *power supply linier*. Tampilan *power supply* dapat dilihat pada gambar 2.22.

Gambar 2.22 Tampilan *Power Supply*

(Sumber : http://skemaku.com)

2.8 *Limit Swicth*

Limit switch adalah alat yang berfungsi untuk menghubungkan dan memutus arus listrik pada suatu rangkaian elektronika, *limit switch* itu memiliki tiga buah terminal, yang pertama adalah *normally close* yang kesua adalah *normally open* dan yang ketiga adalah *central*. *Limit switch* digunakan untuk membatasi kerja dari suatu alat yang sedang beroperasi. NO dan NC serta *Central* dapat di gunakan untuk memutus aliran listrik pada suatu rangkaian atau dapat menyambungkan aliran listrik pada suau rangkaian. Tampilan simbol dan bentuk fisik *limit switch* dapat dilihat pada gambar 2.23.



Gambar 2.23 Tampilan (a) Simbol dan (b) Bentuk fisik *Limit Switch*

(Sumber : http://elektronika-dasar.web.id)

BAB III

PERANCANGAN ALAT DAN PEMBAHASAN

3.1 Tempat dan Waktu Penulisan

Penulisan ini dilakukan di laboraturium teknik elektronika, Fakultas Teknik, Jurusan Teknik Elektronika, UNJ. Waktu penelitian dimulai pada awal semester ganjil 105, tahun akademik 2016/2017.

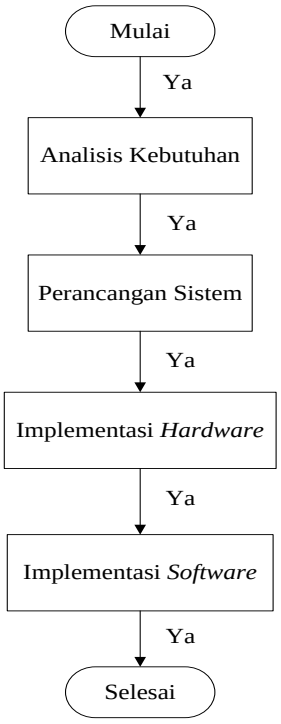
3.2 Alat dan Bahan Penulisan

Dalam penulisan ini, penulis menggunakan bahan dan alat :

1. Laptop yang digunakan dalam penulisan adalah dengan spesifikasi :
   a. *Intel^{R} Core™ i3 CPU 1.70 GHz – 1.7 GHz.*
   b. *Memory 4.00 GB.*
   c. *Hard Disk Drive 500GB.*
   d. *LCD "14"*
   e. *Windows 7, 64 Bit.*
2. Perangkat lunak yang digunakan :
   a. *Arduino,* digunakan untuk memprogram mikrokontroler.
   b. *Eagle 6.1.0*, digunakan untuk membuat skematik dan *layout* rangkaian.
   c. *Google Sketchup Pro 8*, digunakan untuk membuat design.
   d. *Paint*, digunakan untuk membuat gambar perancangan gambar.
   e. *Snipping Tool*, digunakan untuk menggambil gambar pada layar.
   f. *EasyVR Commander*, digunakan untuk menyimpan rekaman suara.
   g. *Proteus 7*, digunakan untuk mensimulasikan rangkaian.
   h. *Microsoft Visio 2007*, digunakan untuk membuat flowchart.

3.3 Diagram Alir Penulisan

Dalam langkah-langkah melakukan penulisan, yang pertama tentukan metode yang akan digunakan dalam proses penulisan. Cara penulisan yang digunakan dalam menyelesaikan penulisan ini adalah cara penelitian dan pengembangan yang meliputi perencanaan, analisis kebutuhan, perancangan, pengujian, dan implementasi sistem. Cara yang digunakan pada penelitian ini dapat dilihat pada gambar 3.1 dengan tahapan seperti gambar di bawah ini.



Gambar 3.1. Tampilan Tahapan Metode Penulisan

Dimana tahapan metode penulisan diawali dengan mulai selanjutnya dengan analisis kebutuhan yaitu menganalisis kebutuhan apa saja yang di butuhkan dalam pembuatan Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino.* Selanjutnya tahapan perancangan sistem dimana membuat rancangan untuk dapat mempermudah

dalam pembuatan Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino.* Setelah itu tahapan pengujian dimana penguji menguji dengan menggunakan sebuah simulasi yaitu *software proteus 7* agar mendapatkan hasil yang sesuai dengan apa yang diinginkan. Selanjutnya tahapan *hardware* yaitu tahapan membuat perangkat keras seperti, tempat helm , *conveyor* dan sebagainya. Tidak lupa yang terakhir adalah tahapan *software* dimana tahapan ini membuat suatu program baik pada *EasyVR* atau *Arduino*. Untuk lebih jelasnya tahapan metode penulisan akan dijelaskan di bawah ini.

3.3.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem adalah tahapan dimana peneliti menentukan kebutuhan dari sistem agar pengembangan Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino* dapat difungsikan sesuai dengan tujuan penulisan Untuk memenuhinya penulis menggunakan *EasyVR* dan Sensor Benda serta *Driver* motor.

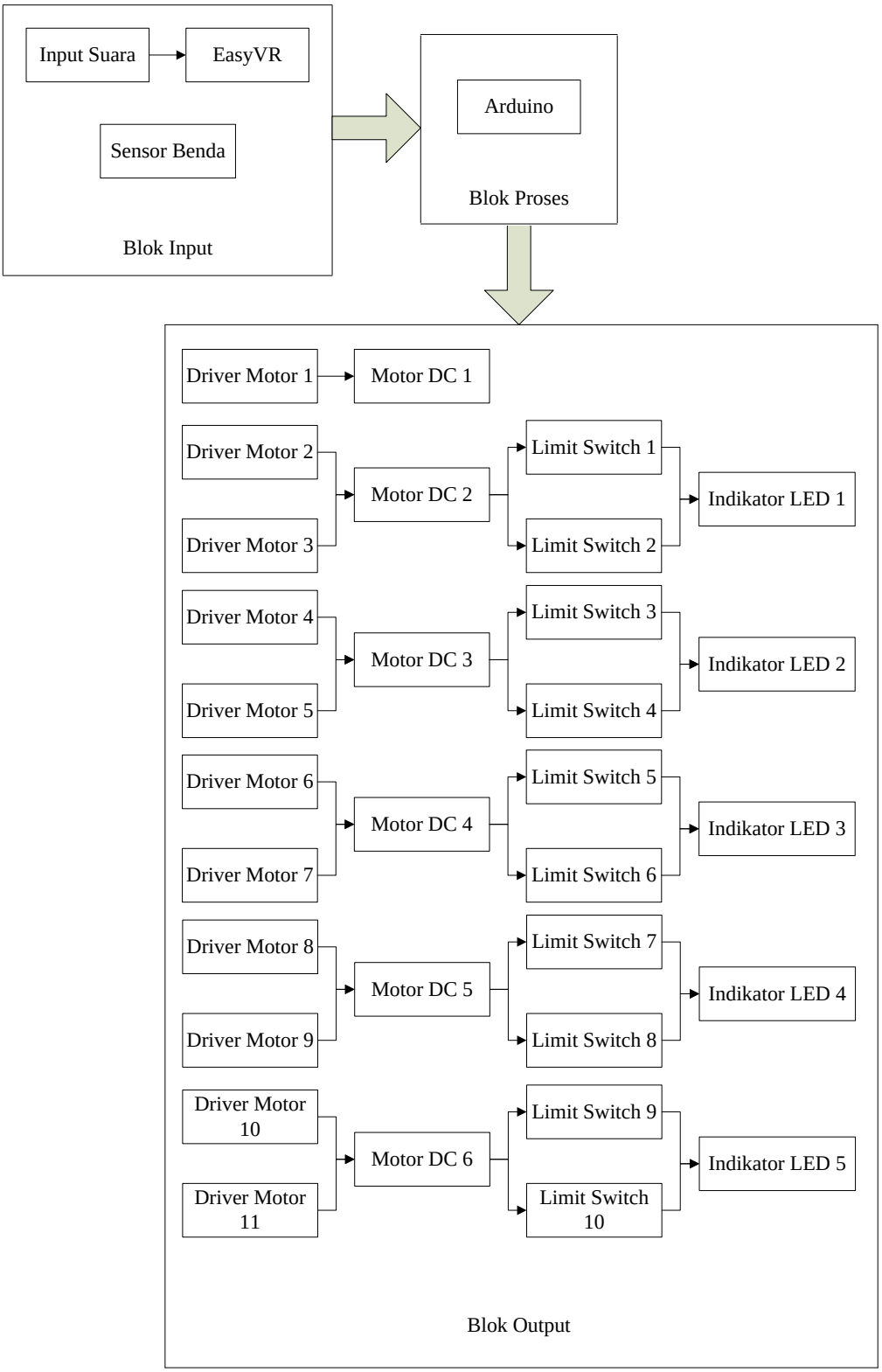
Menganalisa perangkat proses yang digunakan pada sistem, penulisan menggunakan *EasyVR* sebagi perekam dan pemanggil helm, *Arduino* sebagai *converter* antara *EasyVR* dan sensor benda dengan *driver* motor, dan *driver* motor sebagai pengendali motor.

3.3.2 Perancangan Sistem

Perancangan sistem adalah membuat suatu sistem langkah yang tepat sehingga menghasilkan sistem yang baik. Perancangan alat digunakan untuk menentukan komponen dari suatu alat yang digunakan, sehingga hasil akhirnya sesuai dengan yang diharapkan. Perancangan memudahkan peneliti dalam proses

pembuatan alat, dalam perancangan terdiri dari pembuatan diagram blok dan sketsa rangkaian untuk setiap bloknya.

Perancangan sistem pada penulisan ini adalah membangun sebuah sistem blok alat yang dapat menggerakan atau memanggil helm secara otomatis dengan menggunakan sensor-sensor *EasyVR* dan sensor benda yang telah diidentifikasikan dan bagaimana memproses informasi yang didapat sensor menjadi informasi yang dipercaya hasilnya. Tampilan blok diagram perancangan tempat penitipan helm otomatis menggunakan *Voice Recognition* berbasis *Arduino*, dimana dapat dilihat pada gambar 3.2.



Gambar 3.2 Tampilan Blok Diagram Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*.

1. Sensor *EasyVR* adalah sensor yang digunakan untuk menaru dan memanggil helm pada kotak dengan perantara *conveyor*.
2. Sensor benda adalah sensor yang digunakan untuk mendeteksi ada atau tidaknya helm pada kotak penyimpanan. Serta digunakan untuk mengaktifkan motor penarik helm.
3. A*rduino* adalah mikrokontroler yang digunakan untuk menjembatani antara *EasyVR* dan sensor benda dengan *driver* motor.
4. *Driver* motor adalah *driver* yang digunakan untuk mengendalikan motor DC yang mempunyai tegangan DC yang besar.
5. *Limit Switch* digunkan untuk menghentikan gerakan atau putaran pada motor DC.
6. Motor DC adalah motor yang dipakai untuk menggerakan *conveyor* dan untuk menarik helm kedalam kotak penyimpanan serta mendorong helm keluar dari kotak penyimpanan.
7. Indikator LED adalah tanda berupa LED yang menyanya, fungsinya untuk menandakan ada atau tidaknya helm pada kotak penyimpanan.

3.3.3 Implementasi Sistem *Hardware*

Sesudah tahapan pengujian maka tahapan yang selanjutnya adalah membuat *hardware*nya. Design maket dibuat dari bahan triplek serta kayu. Selain sensor-sensor, pada maket juga sudah terpasang motor DC untuk menggerakan helm serta untuk mendorong helm masuk serta keluar dan LED indikator untuk menandakan ada atau tidaknya helm pada tempat penyimpanan.

Perancang perangkat keras menentukan keberhasilan untuk kerja alat Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*.

3.3.4 Implementasi Sistem *Software*

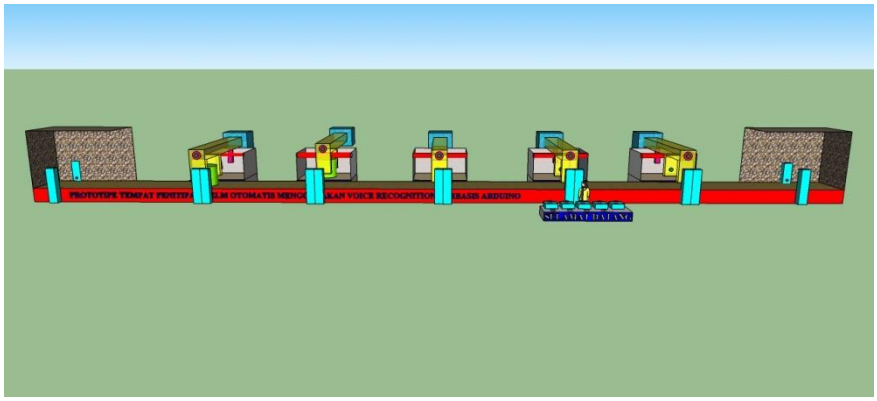
Sistem perangkat lunak atau *software* adalah pembuatan program baik pada *EasyVR* atau *Arduino*. Pada proses ini pembuatan program *EasyVR* harus diperhatikan secara detail dalam pembuatan program agar pada saat upload data tidak ada ke gagalan serta kesalahan dalam menjalankan penitipan helm. Untuk *Arduino* pembuatan program juga harus di perhatikan karena pada *Arduino* berfungsi sebagai penjembatani antara *EasyVR*, sensor benda serta *driver* motor, sehingga sangat penting perannya.

3.3.5 Perancangan Maket Alat

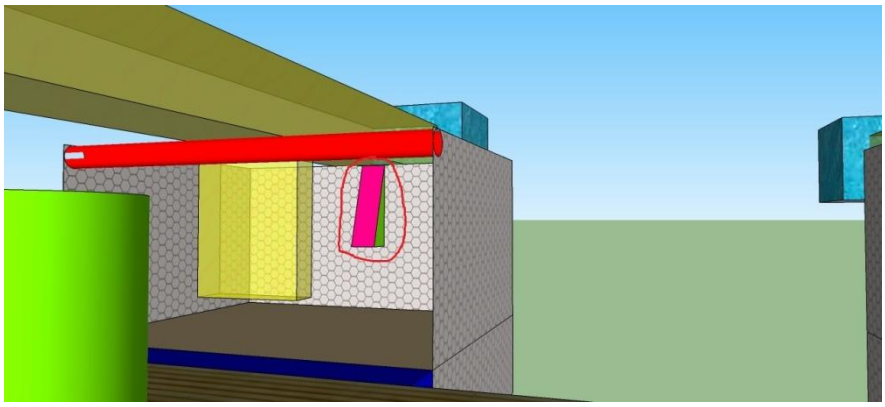
Maket dibuat dari kayu dan triplek sebagai bahan untuk pembuatan kotak helm dan *conveyor*. Penggunaan kayu dan triplek dipilih karena mudah dalam perancangan, serta memudahkan peneliti untuk membuat maket. Maket berupa persegi panjang, segi empat, dan di dalam kotak juga di siapkan motor DC serta *switch saklar*. Perancangan maket alat tempat penitipan helm otomatis dibuat untuk memudahkan uji simulasi dalam menaruh dan mengambil helm.

Maket adalah rancangan arsitektur sebagai cara untuk menyampaikan ide dan menggambar tata ruang. Tujuan membuat maket adalah untuk menguji kualitas rancangan dalam ukuran kecil dan membantu perancang dalam mengembangkan alatnya. Fungsi Sebuah maket adalah digunakan untuk menampilkan dari keadaan yang sesungguhnya menuju kepada keadaan yang akan

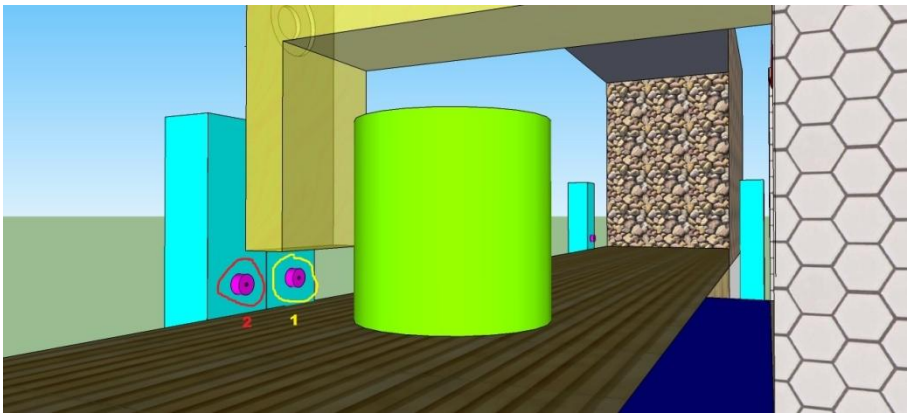
diciptakan. Tampilan perancangan maket alat tempat penitipan helm otomatis menggunakan *Voice Recognition* berbasis *Arduino* dapat dilihat pada gambar 3.3.



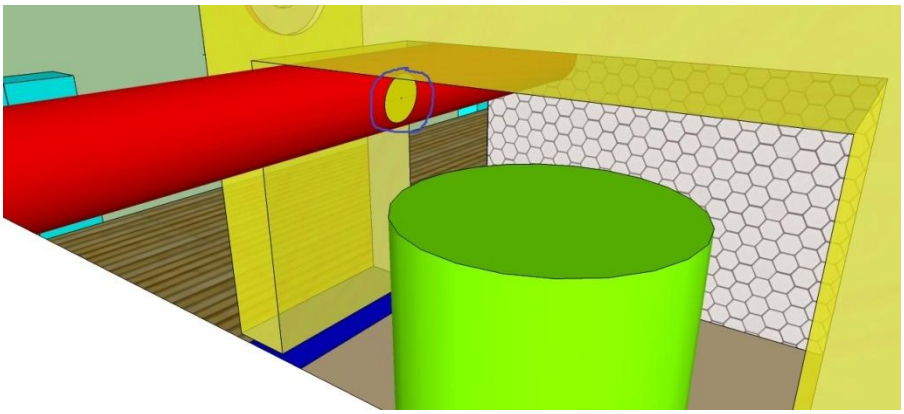
(a)



(b)



(c)



(d)

Gambar 3.3 Tampilan Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*.

Dimana : Gambar (a) gambar keseluruhan dari perancangan maket alat tempat penitipan helm otomatis menggunakan *Voice Recognition* berbasis *Arduino*. Gambar (b) gambar tempat atau posisi *limit switch* 1 fungsinya untuk menghentikan atau mematikan putaran motor penarik helm. Gambar (c) ada dua gambar 1 dan 2 gambar yang nomor 1 adalah sensor untuk penarik helm agar dapat menarik helm masuk ke dalam kotak dan gambar yang nomor 2 adalah sensor untuk mengaktifkan *conveyor* agar motor *conveyor* dapat berjalan membawa helm ke tempat pengambilan helm. Gambar (d) adalah gambar tempat atau posisi *limit switch* 2 fungsinya untuk menghentikan atau mematikan motor pendorong helm.

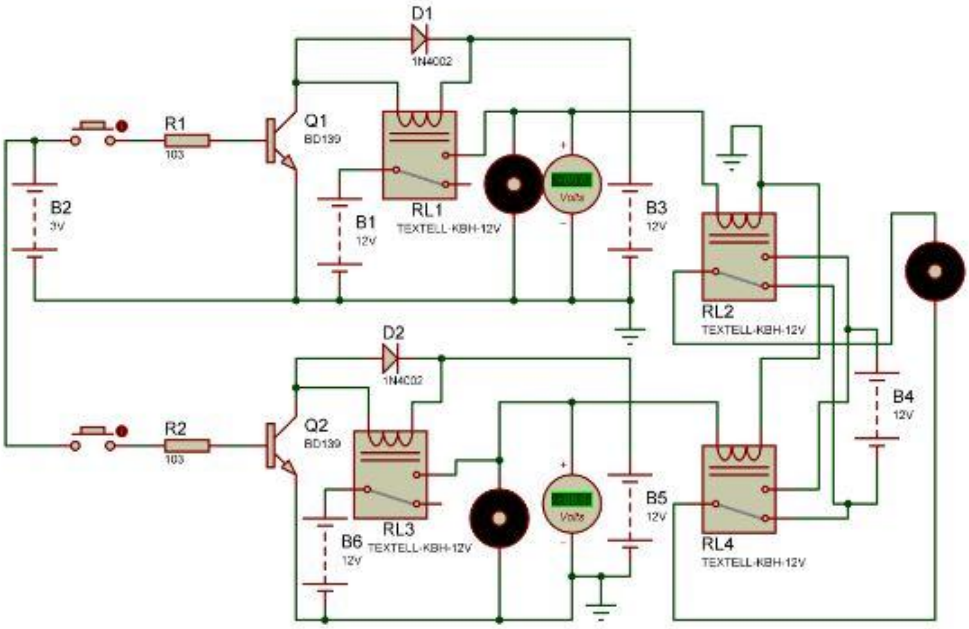
3.3.6 Perancangan Alat

3.3.6.1 Rangkaian *Driver* Motor

Rangkaian sensor *driver* motor untuk menggerakan motor maju dan mundur. Tujuan *driver* motor adalah untuk menguatkan tegangan karena motor

DC membutuhkan tegangan yang lebih besar yang tidak dapat di*supply* oleh tegangan dari keluaran mikrokontroler *Arduino*.

Tampilan rangkaian driver motor dapat dilihat pada gambar 3.4.



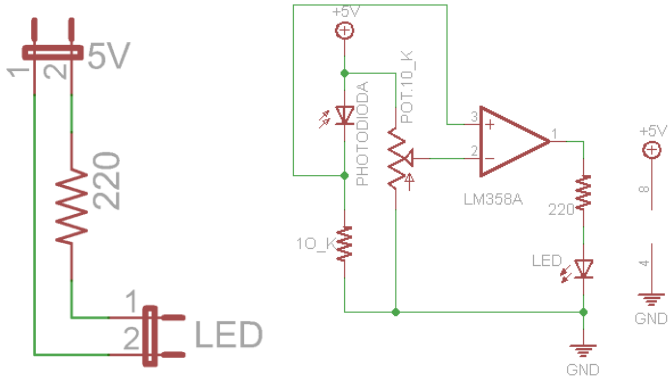
Gambar 3.4 Tampilan Rangkaian *Driver* Motor

Dimana cara kerja *driver* motor adalah keluaran pada *Arduino* yang masuk melalui kaki basis transistor BD 139. Apabila BD 139 mendapat sulutan, maka transistor BD 139 akan terjadi saturasi sehingga relay akan aktif. Ketika common mendapatkan +12V dan apabila BD 139 mendapatkan sulutan, maka transistor BD 139 akan terjadi saturasi. Pada saat terjadi saturasi, maka kontak relay1 yang sebelumnya NC akan menjadi NO, sehingga relay1 aktif dan dapat memberikan input tegangan ke relay2. Pada saat relay2 NO nya medapat +12V, lalu common nya medapat +12V dancoil pada relay2 mendapatkan input tegangan dari NO relay1 maka, kontak relay2 yang tadinya NC akan berubah

menjadi NO sehingga relay2 dapat menggerakan putaran motor ke arah kanan. Untuk putaran motor ke arah kiri cara kerjanya sama seperti cara kerja diatas.

3.3.6.2 Rangkaian Sensor Benda

Rangkaian sensor benda adalah sensor yang di gunakan untuk mendeteksi akan adanya benda yang berada didepan sensor. Sensor benda menggunakan beberapa komponen yaitu LED, IC LM324, Resistor, Trimpot. Rangkaian Sensor Benda digunakan untuk mendeteksi ada atau tidaknya helm. Tampilan sensor benda dapat dilihat pada gambar 3.5.



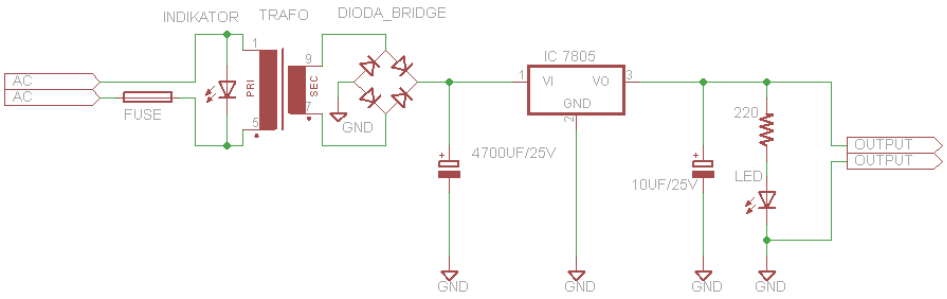
Gambar 3.5 Tampilan Sensor Benda

Apabila Photodioda terkena cahaya yang berasal dari pancaran LED maka hambatan Photodioda akan mengecil. Apabila cahaya LED yg dipancarkan ke Photodioda terpotong atau terhalang oleh sebuah benda, akibatnya Photodioda tidak menerima cahaya sehingga hambatan Photodioda bernilai tinggi.

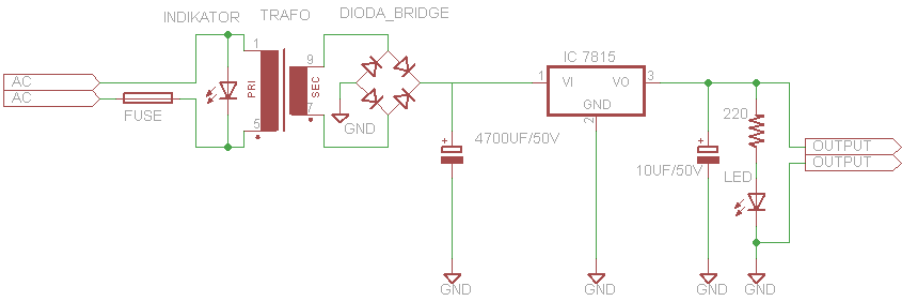
3.3.6.3 Rangkaian *Power Supply*

Rangkaian *regulator* atau *power supply* digunakan sebagai penyuplay tegangan untuk semua rangkaian. Rangkaian *regulator* mendapat tegangan input

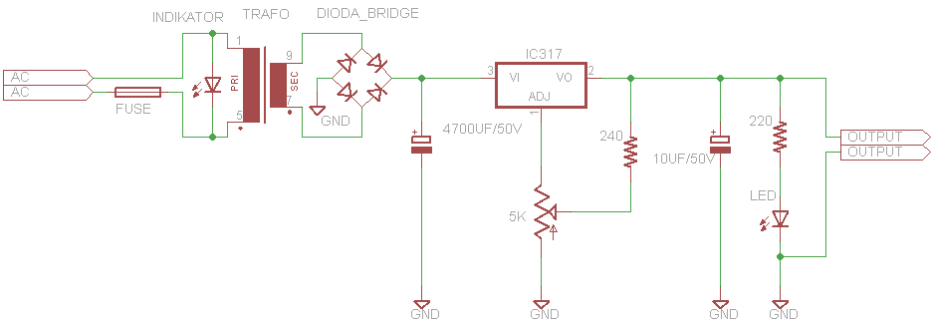
32Volt dari trafo yang kemudian disearahkan menggunakan dioda. Rangkaian *regulator* menggunakan IC *regulator* 7805 yang memberikan tegangan output sebesar 5Volt, IC *regulator* 7815 yang memberikan tegangan output sebesar 15Volt, dan IC LM317 yang memberikan tegangan output sebesar 30Volt. Tampilan rangkaian *power supply* dapat dilihat pada gambar 3.6.



(a)



(b)

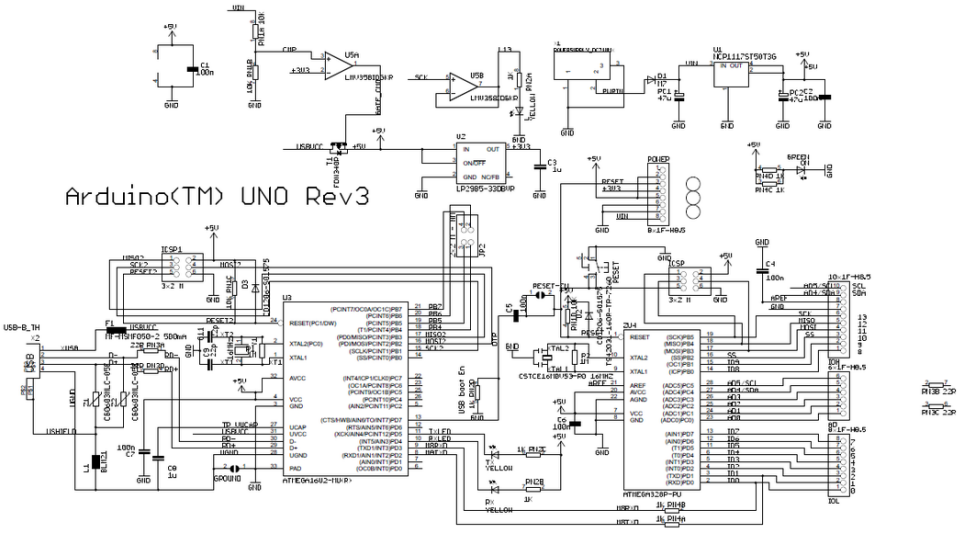


(c)

Gambar 3.6 Tampilan Rangkaian *Power Supply* (a) 5 Volt, (b) 15 Volt, dan (d) 30 Volt

3.3.6.4 *Arduino Uno*

Arduino Uno memikili komunikasi USB, Sehingga pengguna laptop yang tidak memiliki *port* serial dapat menggunakannya. *Arduino Uno* mempunyai modul siap pakai (*Shield*) yang dapat disambungkan pada *Arduino*. *Arduino* berfungsi sebagai pemprosesan data yang dapat di program sesuai dengan program yang telah kita buat, input dari *Arduino* tersebut berasal dari sinyal suara yang berasal dari *microphone* dan telah diproses terlebih dahulu oleh *EasyVR* sebagai *Voice Recognition*. Pada hasil sinyal keluaran pada *Arduino Uno*, sinyal tersebut akan di lanjutkan untuk dapat masuk sebagai input pada *Arduino Mega5260*. Tampilan *Arduino Uno* dapat dilihat pada gambar 3.7.

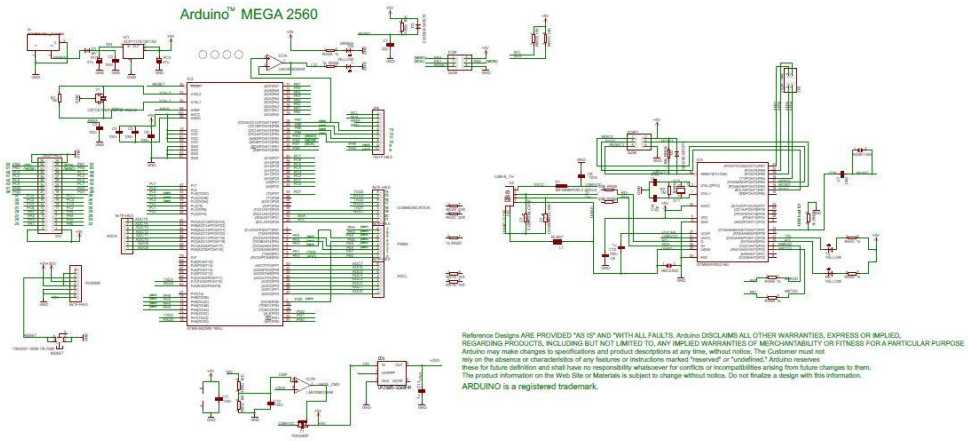


Gambar 3.7 Tampilan *Arduino Uno*

(Sumber : www.veear.eu)

3.3.6.5 *Arduino Mega*

Secara umum *Arduino Mega* memiliki fungsi memudahkan pengguna dalam berbagai bidang elektronik seperti pembuatan aplikasi *running* LED, mobile robot dan masih banyak lagi. Dengan menggunakan *Arduino*, pemakai aplikasi-aplikasi tersebut menjadi lebih mudah dan murah. *Arduino Mega* adalah *Hardware Open Source*, dengan demikian pengguna *Arduino* diberi kebebasan untuk dapat membuat sendiri *Arduinonya*. *Arduino Mega* memiliki *polyfuse reset* yang melindungi *port* USB komputer anda dari konslet dan arus yang berlebihan. *Arduino Mega* dapat diaktifkan melalui koneksi USB atau dengan catu daya eksternal. Tampilan *Arduino Mega* dapat dilihat pada gambar 3.8.



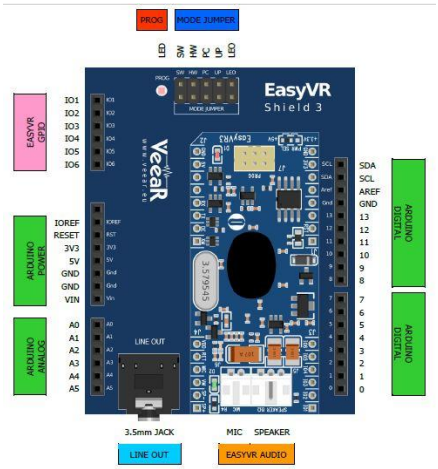
Gambar 3.8 Tampilan *Arduino Mega*

(Sumber : www.veear.eu)

Arduino Mega memiliki 54 pin digital input/output, dimana 15 pin dapat digunakan sebagai output PWM, 16 pin sebagai input analog, dan 4 pin sebagai UART (port serial hardware), 16 MHz Kristal osilator, koneksi USB, jack power header ICSP, dan tombol reset.

3.3.6.6 *EasyVR (Voice Recognition)*

EasyVR merupakan generasi penerus setelah kesuksesan generasi pertama, yaitu *VRBot*. *VRBot* dapat dihubungkan dengan mikrokontroler *Arduino*. Sangat sesuai digunakan dalam berbagai aplikasi contohnya *Home Automation* dimana kita dapat mengontrol nyala lampu, pintu, televisi dan perangkat lainnya. *EasyVR* sudah dapat dipasang langsung pada minimum sistem *Arduino* dengan menggunakan *board EasyVR shield*, karena board ini memang berfungsi untuk menghubungkan *EasyVR* ke macam-macam model *Arduino*. Pada *EasyVR shield* sudah dihubungkan ke pin 12 (*default*) dan pin 8 pada *Arduino* sebagai serial *receiver* (RX) pin yang berfungsi sebagai penerima sinyal serial *(receiver)* serta pin 13 *(default)* dan pin 9 pada *Arduino* sebagai serial *transmitter* (TX) pin yang berfungsi sebagai pemancar sinyal serial (*transmitter)*. Gambar 3.6 merupakan gambar *EasyVR* berserta koneksi pin saat sudah terhubung dengan *Arduino* menggunakan *EasyVR shield*. Tampilan *EasyVR* yang sudah terhubung dengan *Arduino* dapat dilihat pada gambar 3.9.



Gambar 3.9 Tampilan *EasyVR* Yang Terhubung Dengan *Arduino*

(Sumber : www.veear.eu)

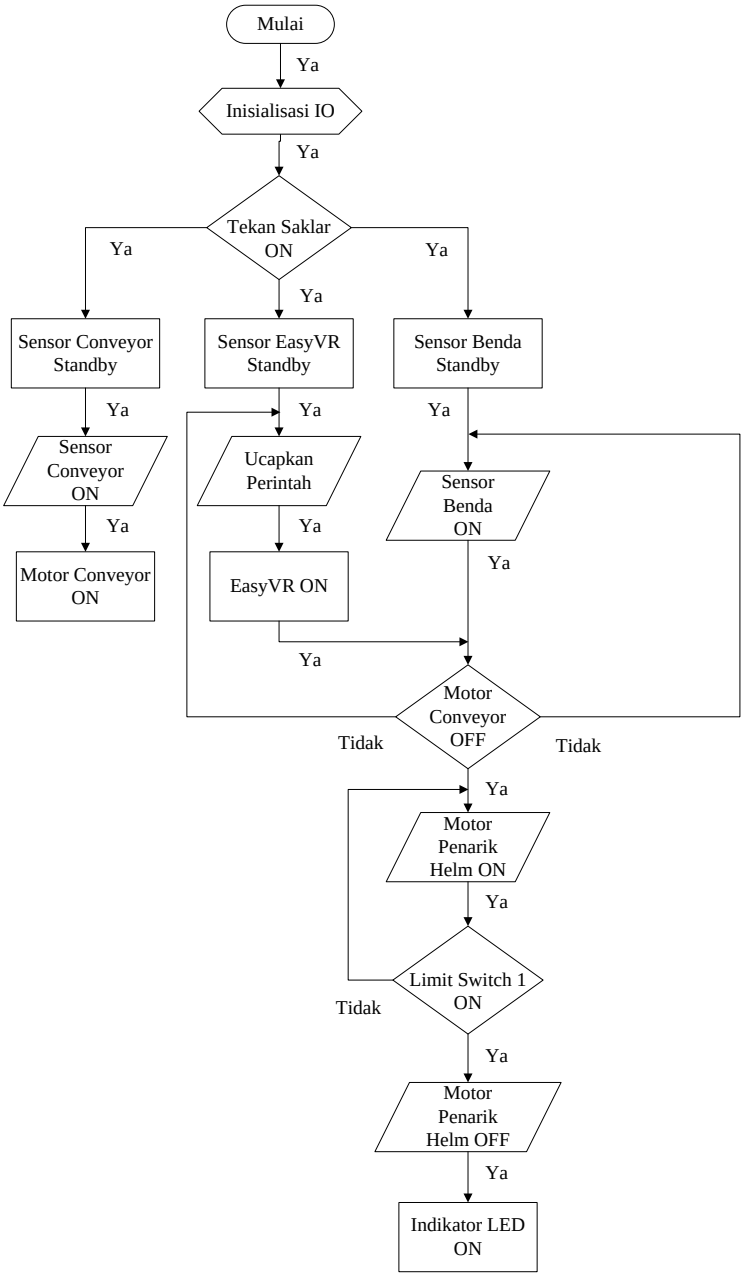
Tabel 3.1 Pin EasyVR Yang Terhubung Dengan *Arduino*

Group	Pin	Description
ARDUINO HEADERS	-	Arduino UNO-R3 Shield interface, pass-through connectors (Pins 0-1 are in use when J12 is set to UP, PC, HW or LEO) (Pins 12-13 or 8-9 are in use when J12 is set to SW)
EASYVR AUDIO	-	Audio cables connectors of the EasyVR 3 module (microphone and speaker)
LINE OUT	-	3.5mm stereo/mono jack (16Ω - 32Ω headphones or line-level output)
MODE JUMPER	SW	Arduino Software Serial (connected to pins 12-13 or 8-9)
	HW	Arduino Hardware Serial (connected to pins 0-1)
	PC	PC Mode (Arduino disabled, EasyVR in command mode)
	UP	Update Mode (Arduino disabled, EasyVR in boot mode)
	LEO	Leonardo Update (Arduino enabled, EasyVR in boot mode)
PROG	-	Red light indicator for Flash programming modes (UP and LEO)
SW SERIAL PINS	RX	Use resistor to select Software Serial RX pin: 12 or 8
	TX	Use resistor to select Software Serial TX pin: 13 or 9
EASYVR GPIO	IO1	General purpose I/O as found on the embedded EasyVR 3 module (referenced at the internal VDD logic level - see note below)
	IO2	
	IO3	
	IO4	
	IO5	
	IO6	

3.3.7 Perancangan *Software*

3.3.7.1 Flowchart Alur Kerja Alat

Dalam pembuatan perancangan perangkat lunak terlebih dahulu dibuat alur kerja alat. Caranya sebagai berikut. Cara pertama adalah sistem penitipan helm. Tampilan sistem penitipan helm dapat dilihat pada gambar 3.10.

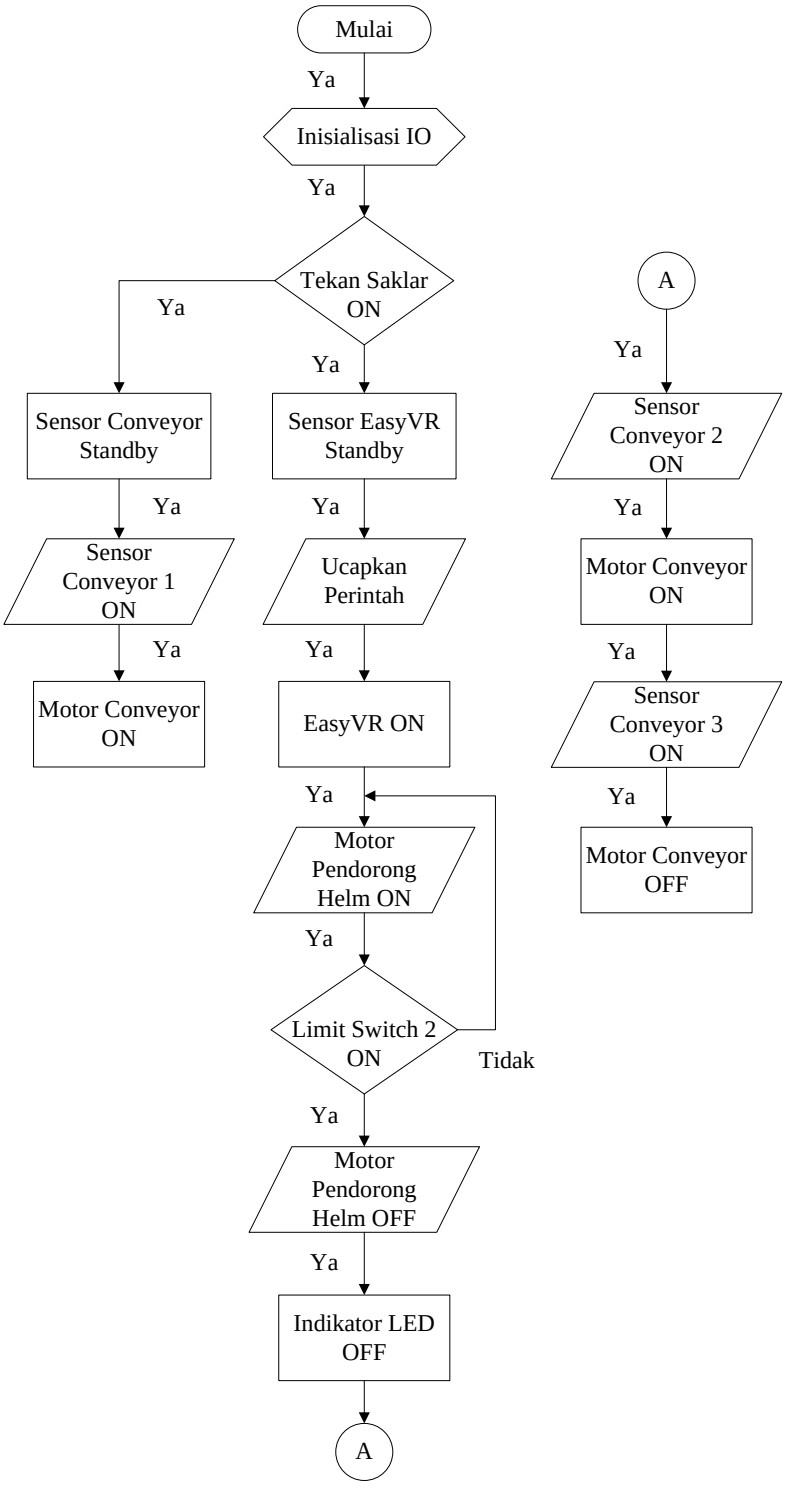


Gambar 3.10 Tampilan Sistem Penitipan Helm

Cara penitipan helm diawali menekan saklar dimana jika saklar dalam keadaan on maka sistem akan aktif, jika tidak maka sistem tidak akan aktif. Setelah saklar on, semua sistem dalam keadaan *standby*, jika sensor *conveyor* on maka motor *conveyor* on. Jika perintah diucapkan pada *EasyVR* maka *EasyVR* on, catatan *EasyVR* akan on jika sudah di simpan terlebih dahulu program suara. Jika

EasyVR merespon panggilan dan sensor benda on, jika tidak maka sistem kembali kepada ucapkan perintah dan sensor benda on. Jika *EasyVR* dan sensor benda on maka, motor penarik helm on. Jika ya maka *limit swicth* 1 on. Jika tidak maka motor penarik helm akan on. Jika *limit swicth* 1 on maka motor penarik helm akan off dan jika ya maka indikator LED on.

Cara kedua adalah sistem pengambilan helm. Tampilan sistem pengambilan helm dapat dilihat pada gambar 3.11.



Gambar 3.11 Tampilan Sistem Pengambilan Helm

Cara Pengambilan helm diawali dengan tekan saklar on. Sensor *Conveyor* dan sensor *EasyVR* akan *standby*. Jika sensor *conveyor* 1 on maka motor *conveyor* on. Jika *EasyVR* medapat perintah ucapan maka *EasyVR* akan on, jika (Ya) maka motor pendorong helm on. Jika motor pendorong helm on (Ya) maka *limit switch* 2 on, jika tidak maka motor pendorong helm akan on. Jika *limit switch* 2 on (Ya) maka pendorong helm off. Setelah itu jika pendorong helm off (Ya) maka indikator LED off. Setelah itu jika indikator LED off (Ya) sensor *conveyor* 2 on, jika (Ya) motor *conveyor* on, jika (Ya) sensor *conveyor* 3 on, jika (Ya) motor *conveyor* off.

3.3.7.2 Cara Kerja Alat

Pada saat helm di letakan pada *conveyor* dan sensor tertutupi dengan helm maka *conveyor* akan berjalan. Jika sensor benda dan *EasyVR* aktif maka helm akan berhenti pas disensor dan helm akan ditarik kedalam tempat penyimpanan, Jika hanya salah satu yang aktif maka helm tidak akan dapat ditarik ke dalam tempat penyimpanan dan *conveyor* pun tidak akan berhenti. Dimana cara kerja *EasyVR* yaitu pengguna cukup mengatakan namanya atau kode yang telah dibuat sebelumnya, jika sistem *EasyVR* mempunyai *sample* suara yang cocok dengan penguna maka pengguna akan diakui sebagai pengguna yang *valid* dari sistem. Sensor helm akan aktif, jika sensor helm terhalangi oleh helm, jika sensor helm aktif maka motor *conveyor* akan berhenti dan motor penarik helm akan aktif. Motor penarik helm akan berhenti apabila *limit switch* 1 tertekan dan indikator adanya helm pada kotak penyimpanan akan menyala. Setelah itu penjaga

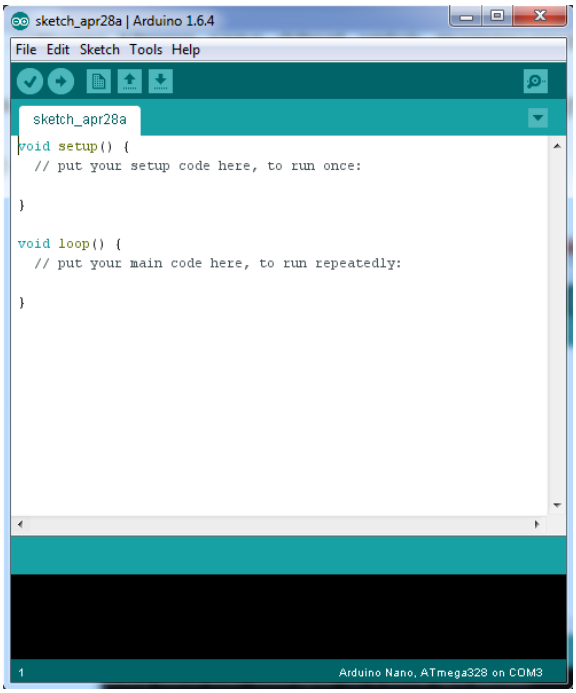
akan memberikan sebuah kartu yang menandakan tempat nomor penyimpanan helm anda.

Sistem kedua yaitu pada saat konsumen akan mengambil helm, konsumen hanya perlu menyerahkan kartu yang telah diberikan kepada penjaga. Penjaga cukup mengatakan namanya atau kode yang telah di buat sebelumnya pada mikrofon *EasyVR*. Jika *valid* motor pendorong helm akan aktif, *Jika limit switch* 2 tertekan maka motor pendorong helm akan berhenti dan jika motor pendorong helm berhenti, menandakan helm sudah berada diluar tempat penyimpanan helm dan helm sudah berada di jalur *conveyor*. Dijalur *conveyor* sudah terdapat sensor utuk mengaktifkan *conveyor* maka *conveyo*r akan berjalan kembali dan membawa helm kepada konsumen. Helm akan berhenti jika helm mengenai sensor pemberhenti *conveyor*.

3.3.7.3 Program *Arduino*

Peneliti menggunakan *software Arduino* untuk membuat program Perancangan Tempat Penitipan Helm Otomatis Menggunakan *Voice Recognition* Berbasis *Arduino*.

Alasan penulis menggunakan *software* ini karena *software* ini merupakan *software* yang dibuat untuk membuat program semua jenis *Arduino* yang ada. Tampilan awal *software Arduino* dapat dilihat pada gambar 3.12.



Gambar 3.12 Tampilan Awal *Software Arduino*

Pada alat Perancangan Tempat Penitipan Helm Otomatis Menggunakan Voice Recognition Berbasis *Arduino* digunakan beberapa input dan output. Berikut ini adalah keterangan pin yang digunakan pada *Arduino Uno* dan *Arduino Mega*.

1. Pin yang digunakan pada *Arduino Uno* dapat dilihat pada Tabel 3.2

Tabel 3.2 Pin *Arduino Uno*

No	Input	Output
1	Sensor *EasyVR*	*8/12 (RX)* dan 9/13 (*TX)*

2. Pin yang digunakan pada *Arduino Mega*, dapat dilihat pada Tabel 3.3

Tabel 3.3 *Arduino Mega*

No	Input	Output	Kotak Tempat Helm
1	A0 sampai dengan A7	2, 3 dan 4	Kotak 1
2	A8 sampai dengan A13	5 dan 6	Kotak 2
3	A14, A15, 30 sampai dengan 33	7 dan 8	Kotak 3
4	34 sampai dengan 39	9 dan 10	Kotak 4
5	40 sampai dengan 45	11 dan 12	Kotak 5

3. Output yang dipakai pada Arduino Uno dapat dilihat pada Tabel 3.4

Tabel 3.4 *Arduino Uno*

No	Output	Fungsinya
1	2 dan 3	Untuk keluar helm dan masuk helm
2	4 dan 5	Untuk keluar helm dan masuk helm
3	6 dan 7	Untuk keluar helm dan masuk helm
4	10 dan 11	Untuk keluar helm dan masuk helm
5	A4 dan A5	Untuk keluar helm dan masuk helm

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Dari hasil penulisan yang telah dilakukan dapat diambil beberapa kesimpulan :

1. Potodioda dan LED dapat digunakan sebagai sensor benda dimana LED difungsikan sebagai *transmitter* dan potodioda sebagai *receiver*. Data keluaran sensor benda berupa nilai tegangan. Nilai tegangan ini akan diolah di dalam mikrokontroller *Arduino Mega*. Untuk mengendalikan putaran motor DC dapat menggunakan sebuah *driver* motor DC. Dimana fungsi dari driver DC adalah untuk menguatkan tegangan motor DC karena motor DC membutuhkan tegangan yang lebih besar yang tidak dapat di*supply* oleh tegangan dari keluaran mikrokontroler *Arduino*.
2. Mikrokontroler *Arduino Uno* berfungsi sebagai pemprosesan data yang dapat di program sesuai dengan program yang telah kita buat, input dari arduino tersebut berasal dari sinyal suara yang berasal dari *microphone* dan telah diproses terlebih dahulu oleh *EasyVR* sebagai *Voice recognition*. Pada hasil sinyal keluaran pada *Arduino Uno* sinyal tersebut akan di lanjutkan untuk dapat masuk sebagai input pada *Arduino Mega*.
3. Mikrokontroler *Arduino Mega* berfungsi sebagai jembatan antara sensor benda dengan *Arduino Uno* sehingga program yang dibuat dapat berjalan sesuai seperti apa yang di inginkan.
4. *EasyVR* adalah merupakan modul *Voice Recognition* dengan fungsi yang banyak. *EasyVR* banyak digunakan pada aplikasi pengontrolan yang

membutuhkan pendeteksian suara. *EasyVR* digunakan sebagai sensor untuk menggerakan motor.

5. Rangkaian *regulator* atau *power supply* digunakan sebagai penyuplay tegangan untuk semua rangkaian. *Power Supply* atau disebut dengan catu daya adalah suatu alat listrik yang dapat memberikan energi listrik untuk pelektronika lainnya. Pada dasarnya *Power Supply* atau catu daya memerlukan sumber energi listrik yang kemudian mengubahnya menjadi energi listrik yang dibutuhkan oleh perangkat elektronika lainnya.

4.2 Saran

Untuk penelitian atau pembuatan alat ini selanjutnya peneliti menyarankan :

1. Dalam penggunaan *EasyVR* keadaan sekitar harus dalam keadaan hening.
2. Jarak antara *microphone* dengan bibir pengguna haruslah pas atau sesuai dengan jarak yang di tentukan.
3. Ucapan bahasa atau program haruslah sama dengan apa yang di rekam sebelumnya pada *EasyVR*.
4. LED pada sensor benda dan sensor *conveyor* haruslah terang, karena jika LED tidak terang maka sensor tidak akan berfungsi dengan baik.
5. Motor DC yang digunakan dalam menggerakan *conveyor* dan penarikan serta mendorong helm haruslah bertipe tenaga dan torsi besar.

DAFTAR PUSTAKA

ArduinoMega 2560 datasheet. (n.d.). Arduino Mega2560. *Power*, 1–7. Retrieved from http://www.robotshop.com/content/PDF/ArduinoMega2560Datasheet.pdf

Arifadli. (2011). Sensor Rangkaian Photodioda. Retrieved from https://fadli84.wordpress.com/tag/sensor-rangkaian-photo-dioda/

Chandra MDE. (2010). Rangkaian Driver Relay Praktis Menggunakan Transistor Bipolar. Retrieved from https://telinks.wordpress.com/2010/05/01/rangkaian-driver-relay-praktis-menggunakan-transistor-bipolar/

Djuandi, F. (2011). Pengenalan Arduino. *E-Book. Www. Tobuku*, 1–24. Retrieved from http://www.tobuku.com/docs/Arduino-Pengenalan.pdf

Elektronika. (n.d.). Pengertian Power Supply Jenis Catu Daya. Retrieved from http://teknikelektronika.com/pengertian-power-supply-jenis-catu-daya/

Elektronika. (2016). Pengertian Relay Dan Fungsi Relay.

Elektronika Dasar. (2015). SPST - SPDT - DPDT - 3PDT. Retrieved from http://www.diypedalindonesia.com/spst-spdt-dpdt-3pdt-apaan-sih/

Iksan. (2012). Fungsi dan Kegunaan Conveyor. Retrieved from http://fungsi.info/fungsi-dan-kegunaan-conveyor/

Ismi reza sutanty. (2014). Pengertian Maket. Retrieved from http://ismiartikelbangunan.blogspot.co.id/2014/08/pengertian-maket.html

Kristianto, B. (2013). Motor DC & Motor AC. Retrieved from http://blogs.itb.ac.id/el2244k0112211049briankristianto1/2013/04/27/lalala/

Manual, U. (n.d.). EasyVR 3. Retrieved from https://arduino.cc

Maulid Supriyono. (2013). Prinsip Kerja Limit Switch. Retrieved from https://informasicuy.blogspot.co.id/2013/07/prinsip-kerja-limit-switch.html

Moh Duro. (2012). Fungsi Dan Jenis jenis Relay. Retrieved from http://www.dien-elcom.com/2012/08/fungsi-dan-jenis-jenis-relay.html

Rohmattullah. (2015). Pengertian dan Fungsi Catu Daya Secara Umum. Retrieved from http://rohmattullah.student.telkomuniversity.ac.id/pengertian-dan-fungsi-catu-daya-secara-umum/

Ryan Ferdy Permadi. (2012). Pengertian Photodioda. Retrieved from https://ryankudeta.wordpress.com/2012/12/17/pengertian-photodioda/

Sheva. (2012). Limit Switch Dan Saklar Push On. Retrieved from http://elektronika-dasar.web.id/limit-switch-dan-saklar-push-on/

Sheva Permana. (2013). Definisi Maket. Retrieved from http://teknologikom.blogspot.co.id/2013/04/definisi-maket.html

Suluh. (2012). Anatomi Sistem Roller Conveyor. Retrieved from https://suluhmania.wordpress.com/2012/04/04/anatomi-sistemroller-conveyor/

LAMPIRAN

Lampiran 1

Program pada *Arduino Mega* 1 Kotak

```
// Nomor pada pin input
const int buttonPin = A0;
    // Sensor Benda Untuk Aktifin Motor Conveyor1 (pada saat taru helm)
const int button2Pin = A1; // Sensor Benda Untuk Aktifin Motor Tarik Helm
const int button3Pin = A2; // Saklar Untuk Berhentiin Motor Tarik Helm
const int button4Pin = A3; // Sensor Suara Untuk Aktifin Motor Dorong Helm
const int button5Pin = A4; // Saklar Untuk Berhentiin Motor Dorong Helm
const int button6Pin = A5;
    // Sensor Benda Untuk Berhentiin Motor Conveyor (helm sudah ditempat
       pengambilan helm)
const int button7Pin = A6; // Sensor Suara Untuk Aktifin Motor Tarik Helm
const int button8Pin = A7;
    // Sensor Benda Untuk Aktifin Motor Conveyor2 (pada saat helm keluar
       dari kotak penyimpanan)
// Nomor pada pin output
const int ledPin =  2;  // Motor Conveyor
const int led2Pin = 3; // Motor Tarik Helm
const int led3Pin = 4; // Motor Dorong Helm
```

```
// Kondisi awal variable
int buttonState = 0;   // Kondisi Awal Tombol button = 0
int button2State = 0; // Kondisi Awal Tombol button = 0
int button3State = 0; // Kondisi Awal Tombol button = 0
int button4State = 0; // Kondisi Awal Tombol button = 0
int button5State = 0; // Kondisi Awal Tombol button = 0
int button6State = 0; // Kondisi Awal Tombol button = 0
int button7State = 0; // Kondisi Awal Tombol button = 0
int button8State = 0; // Kondisi Awal Tombol button = 0

void setup()
      // Semua kode didalam kurung kurawal akan dijalankan hanya satu kali
         ketika program Arduino dijalankan untuk pertama kalinya.
{
 // Inisialisasi tombol pin output
 pinMode(ledPin, OUTPUT);
      // Mengatur Digital Pin ledPin Sebagai Output Motor Conveyor
 pinMode(led2Pin, OUTPUT);
      // Mengatur Digital Pin led2Pin Sebagai Output Motor Tarik Helm
 pinMode(led3Pin, OUTPUT);
      // Mengatur Digital Pin led3Pin Sebagai Output Motor Dorong Helm
```

```
 // Inisialisasi tombol pin input
 pinMode(buttonPin, INPUT);   // Tombol Button Sebagai Input
 pinMode(button2Pin, INPUT); // Tombol Button2 Sebagai Input
 pinMode(button3Pin, INPUT); // Tombol Button3 Sebagai Input
 pinMode(button4Pin, INPUT); // Tombol Button4 Sebagai Input
 pinMode(button5Pin, INPUT); // Tombol Button5 Sebagai Input
 pinMode(button6Pin, INPUT); // Tombol Button6 Sebagai Input
 pinMode(button7Pin, INPUT); // Tombol Button7 Sebagai Input
 pinMode(button8Pin, INPUT); // Tombol Button8 Sebagai Input
}

 void loop()
      // Fungsi ini akan dijalankan setelah setup (fungsi void setup) selesai.
      // Setelah dijalankan satu kali fungsi ini akan dijalankan lagi, dan lagi secara
         terus menerus sampai catu daya (power) dilepaskan.
{
 buttonState = digitalRead(buttonPin);     // Baca status dari pin button
 button2State = digitalRead(button2Pin); // Baca status dari pin button 2
 button3State = digitalRead(button3Pin); // Baca status dari pin button 3
 button4State = digitalRead(button4Pin); // Baca status dari pin button 4
 button5State = digitalRead(button5Pin); // Baca status dari pin button 5
 button6State = digitalRead(button6Pin); // Baca status dari pin button 6
 button7State = digitalRead(button7Pin); // Baca status dari pin button 7
```

```
 button8State = digitalRead(button8Pin); // Baca status dari pin button 8
////////////////////////////////////////////////////MASUK/////////////////////////////////////////////////////
  if (buttonState == HIGH){      // Sensor Untuk Aktifin Conveyor
                                 // Jika tombol button = HIGH maka,
    digitalWrite(ledPin, HIGH);   // Mengatur ledPin Agar Motor Conveyor ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button2State == HIGH && button7State == HIGH){
        // Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
        // Jika tombol button 2 = HIGH dan tombol button 7 = HIGH maka,
    digitalWrite(ledPin,LOW); // Mengatur ledPin Agar Motor Conveyor OFF dan
    digitalWrite(led2Pin,HIGH); // Mengatur led2Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button3State == HIGH){    // Saklar Untuk Berhentiin Motor Tarik Helm
                                // Jika tombol button 3 = HIGH maka,
    digitalWrite(led2Pin,LOW); // Mengatur led2Pin Agar Motor Tarik Helm OFF
}
////////////////////////////////////////////////////KELUAR/////////////////////////////////////////////////////
  if (button4State == HIGH){    // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                // Jika tombol button 4 = HIGH maka,
    digitalWrite(led3Pin,HIGH); //Mengatur led3Pin Agar Motor Dorong Helm ON
  }
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button5State == HIGH){    // Saklar Untuk Berhentiin Motor Dorong
                                // Jika tombol button 5 = HIGH maka,
    digitalWrite(led3Pin,LOW);  // Mengatur led3Pin Agar Motor Dorong Helm
                                   OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button8State == HIGH && button5State == HIGH){
        // Sensor Benda dan Saklar Aktifin Motor Conveyor
        // Jika tombol button 8 = HIGH dan tombol button 5 = HIGH maka,
    digitalWrite(ledPin,HIGH);  // Mengatur ledPin Agar Motor Conveyor ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button6State == HIGH){  // Sensor Berhentiin Motor Conveyor
                              // Jika tombol button 6 = HIGH maka,
    digitalWrite(ledPin,LOW); // Mengatur ledPin Agar Motor Conveyor OFF
  }
}
```

Lampiran 2

Program pada *Arduino Mega* Gabungan

```
// Nomor pada pin input

// Kotak 1

const int buttonPin = A0;  // Sensor Benda Untuk Aktifin Motor Conveyor1 (pada saat taru helm)

const int button2Pin = A1; // Sensor Benda Untuk Aktifin Motor Tarik Helm

const int button3Pin = A2; // Saklar Untuk Berhentiin Motor Tarik Helm

const int button4Pin = A3; // Sensor Suara Untuk Aktifin Motor Dorong Helm

const int button5Pin = A4; // Saklar Untuk Berhentiin Motor Dorong Helm

const int button6Pin = A5; // Sensor Benda Untuk Berhentiin Motor Conveyor( helm sudah ditempat pengambilan helm)

const int button7Pin = A6; // Sensor Suara Untuk Aktifin Motor Tarik Helm

const int button8Pin = A7; // Sensor Benda Untuk Aktifin Motor Conveyor2 (pada saat helm keluar dari kotak penyimpanan)

// Kotak 2

const int button9Pin = A8;   // Sensor Benda Untuk Aktifin Motor Tarik Helm

const int button10Pin = A9;  // Saklar Untuk Berhentiin Motor Tarik Helm

const int button11Pin = A10; // Sensor Suara Untuk Aktifin Motor Dorong Helm

const int button12Pin = A11; // Saklar Untuk Berhentiin Motor Dorong Helm

const int button13Pin = A12; // Sensor Suara Untuk Aktifin Motor Tarik Helm

const int button14Pin = A13; // Sensor Benda Untuk Aktifin Motor Conveyor2 (pada saat helm keluar dari kotak penyimpanan)
```

```
// Kotak 3

const int button15Pin = A14; // Sensor Benda Untuk Aktifin Motor Tarik Helm

const int button16Pin = A15; // Saklar Untuk Berhentiin Motor Tarik Helm

const int button17Pin = 30;  // Sensor Suara Untuk Aktifin Motor Dorong Helm

const int button18Pin = 31;  // Saklar Untuk Berhentiin Motor Dorong Helm

const int button19Pin = 32;  // Sensor Suara Untuk Aktifin Motor Tarik Helm

const int button20Pin = 33; // Sensor Benda Untuk Aktifin Motor Conveyor2
                            (pada saat helm keluar dari kotak penyimpanan)

// Kotak 4

const int button21Pin = 34; // Sensor Benda Untuk Aktifin Motor Tarik Helm

const int button22Pin = 35; // Saklar Untuk Berhentiin Motor Tarik Helm

const int button23Pin = 36; // Sensor Suara Untuk Aktifin Motor Dorong Helm

const int button24Pin = 37; // Saklar Untuk Berhentiin Motor Dorong Helm

const int button25Pin = 38; // Sensor Suara Untuk Aktifin Motor Tarik Helm

const int button26Pin = 39; // Sensor Benda Untuk Aktifin Motor Conveyor2
                            (pada saat helm keluar dari kotak penyimpanan)

// Kotak 5

const int button27Pin = 40; // Sensor Benda Untuk Aktifin Motor Tarik Helm

const int button28Pin = 41; // Saklar Untuk Berhentiin Motor Tarik Helm

const int button29Pin = 42; // Sensor Suara Untuk Aktifin Motor Dorong Helm

const int button30Pin = 43; // Saklar Untuk Berhentiin Motor Dorong Helm

const int button31Pin = 44; // Sensor Suara Untuk Aktifin Motor Tarik Helm

const int button32Pin = 45; // Sensor Benda Untuk Aktifin Motor Conveyor2
                            (pada saat helm keluar dari kotak penyimpanan)
```

```
// Nomor pada pin output
const int ledPin =  2; // Motor Conveyor
// Kotak 1
const int led2Pin = 3; // Motor Tarik Helm
const int led3Pin = 4; // Motor Dorong Helm
// Kotak 2
const int led4Pin = 5; // Motor Tarik Helm
const int led5Pin = 6; // Motor Dorong Helm
// Kotak 3
const int led6Pin = 7; // Motor Tarik Helm
const int led7Pin = 8; // Motor Dorong Helm
// Kotak 4
const int led8Pin = 9; // Motor Tarik Helm
const int led9Pin = 10; // Motor Dorong Helm
// Kotak 5
const int led10Pin = 11; // Motor Tarik Helm
const int led11Pin = 12; // Motor Dorong Helm

// kondisi awal variable
// Kotak 1
int buttonState = 0; // Kondisi Awal Tombol button = 0
int button2State = 0; // Kondisi Awal Tombol button = 0
int button3State = 0; // Kondisi Awal Tombol button = 0
```

```
int button4State = 0; // Kondisi Awal Tombol button = 0
int button5State = 0; // Kondisi Awal Tombol button = 0
int button6State = 0; // Kondisi Awal Tombol button = 0
int button7State = 0; // Kondisi Awal Tombol button = 0
int button8State = 0; // Kondisi Awal Tombol button = 0
// Kotak 2
int button9State = 0; // Kondisi Awal Tombol button = 0
int button10State = 0; // Kondisi Awal Tombol button = 0
int button11State = 0; // Kondisi Awal Tombol button = 0
int button12State = 0; // Kondisi Awal Tombol button = 0
int button13State = 0; // Kondisi Awal Tombol button = 0
int button14State = 0; // Kondisi Awal Tombol button = 0
// Kotak 3
int button15State = 0; // Kondisi Awal Tombol button = 0
int button16State = 0; // Kondisi Awal Tombol button = 0
int button17State = 0; // Kondisi Awal Tombol button = 0
int button18State = 0; // Kondisi Awal Tombol button = 0
int button19State = 0; // Kondisi Awal Tombol button = 0
int button20State = 0; // Kondisi Awal Tombol button = 0
// Kotak 4
int button21State = 0; // Kondisi Awal Tombol button = 0
int button22State = 0; // Kondisi Awal Tombol button = 0
int button23State = 0; // Kondisi Awal Tombol button = 0
```

```
int button24State = 0; // Kondisi Awal Tombol button = 0

int button25State = 0; // Kondisi Awal Tombol button = 0

int button26State = 0; // Kondisi Awal Tombol button = 0

// Kotak 5

int button27State = 0; // Kondisi Awal Tombol button = 0

int button28State = 0; // Kondisi Awal Tombol button = 0

int button29State = 0; // Kondisi Awal Tombol button = 0

int button30State = 0; // Kondisi Awal Tombol button = 0

int button31State = 0; // Kondisi Awal Tombol button = 0

int button32State = 0; // Kondisi Awal Tombol button = 0


void setup() // Semua kode didalam kurung kurawal akan dijalankan hanya satu
                kali ketika program Arduino dijalankan untuk pertama kalinya.

{

  // Inisialisasi tombol pin output

  pinMode(ledPin, OUTPUT); // Mengatur Digital Pin ledPin Sebagai Output
Motor Conveyor


  // Kotak 1

  pinMode(led2Pin, OUTPUT); // Mengatur Digital Pin led2Pin Sebagai Output
                              Motor Tarik Helm

  pinMode(led3Pin, OUTPUT); // Mengatur Digital Pin led3Pin Sebagai Output
                              Motor Dorong Helm
```

```
// Kotak 2

pinMode(led4Pin, OUTPUT); // Mengatur Digital Pin led4Pin Sebagai Output
                             Motor Tarik Helm

pinMode(led5Pin, OUTPUT); // Mengatur Digital Pin led5Pin Sebagai Output
                             Motor Dorong Helm

 // Kotak 3

pinMode(led6Pin, OUTPUT); // Mengatur Digital Pin led6Pin Sebagai Output
                             Motor Tarik Helm

pinMode(led7Pin, OUTPUT); // Mengatur Digital Pin led7Pin Sebagai Output
                             Motor Dorong Helm

 // Kotak 4

pinMode(led8Pin, OUTPUT); // Mengatur Digital Pin ledPin Sebagai Output
                             Motor Tarik Helm

pinMode(led9Pin, OUTPUT); // Mengatur Digital Pin led9Pin Sebagai Output
                             Motor Dorong Helm

 // Kotak 5

pinMode(led10Pin, OUTPUT); // Mengatur Digital Pin led10Pin Sebagai Output
                              Motor Tarik Helm

pinMode(led11Pin, OUTPUT); // Mengatur Digital Pin led11Pin Sebagai Output
                              Motor Dorong Helm


// Inisialisasi tombol pin input

// Kotak 1

pinMode(buttonPin, INPUT); // Tombol Button Sebagai Input

pinMode(button2Pin, INPUT); // Tombol Button2 Sebagai Input

pinMode(button3Pin, INPUT); // Tombol Button3 Sebagai Input

pinMode(button4Pin, INPUT); // Tombol Button4 Sebagai Input
```

```
pinMode(button5Pin, INPUT); // Tombol Button5 Sebagai Input
pinMode(button6Pin, INPUT); // Tombol Button6 Sebagai Input
pinMode(button7Pin, INPUT); // Tombol Button7 Sebagai Input
pinMode(button8Pin, INPUT); // Tombol Button8 Sebagai Input
// Kotak 2
pinMode(button9Pin, INPUT); // Tombol Button9 Sebagai Input
pinMode(button10Pin, INPUT); // Tombol Button10 Sebagai Input
pinMode(button11Pin, INPUT); // Tombol Button11 Sebagai Input
pinMode(button12Pin, INPUT); // Tombol Button12 Sebagai Input
pinMode(button13Pin, INPUT); // Tombol Button13 Sebagai Input
pinMode(button14Pin, INPUT); // Tombol Button14 Sebagai Input
// Kotak 3
pinMode(button15Pin, INPUT); // Tombol Button15 Sebagai Input
pinMode(button16Pin, INPUT); // Tombol Button16 Sebagai Input
pinMode(button17Pin, INPUT); // Tombol Button17 Sebagai Input
pinMode(button18Pin, INPUT); // Tombol Button18 Sebagai Input
pinMode(button19Pin, INPUT); // Tombol Button19 Sebagai Input
pinMode(button20Pin, INPUT); // Tombol Button20 Sebagai Input
// Kotak 4
pinMode(button21Pin, INPUT); // Tombol Button21 Sebagai Input
pinMode(button22Pin, INPUT); // Tombol Button22 Sebagai Input
pinMode(button23Pin, INPUT); // Tombol Button23 Sebagai Input
pinMode(button24Pin, INPUT); // Tombol Button24 Sebagai Input
```

```
 pinMode(button25Pin, INPUT); // Tombol Button25 Sebagai Input
 pinMode(button26Pin, INPUT); // Tombol Button26 Sebagai Input
 // Kotak 5
 pinMode(button27Pin, INPUT); // Tombol Button27 Sebagai Input
 pinMode(button28Pin, INPUT); // Tombol Button28 Sebagai Input
 pinMode(button29Pin, INPUT); // Tombol Button29 Sebagai Input
 pinMode(button30Pin, INPUT); // Tombol Button30 Sebagai Input
 pinMode(button31Pin, INPUT); // Tombol Button31 Sebagai Input
 pinMode(button32Pin, INPUT); // Tombol Button32 Sebagai Input


}
 void loop() // Fungsi ini akan dijalankan setelah setup (fungsi void setup) selesai.
        // Setelah dijalankan satu kali fungsi ini akan dijalankan lagi, dan lagi
          secara terus menerus sampai catu daya (power) dilepaskan.
{
 // Kotak 1
buttonState = digitalRead(buttonPin); // Baca status dari pin button
button2State = digitalRead(button2Pin); // Baca status dari pin button 2
button3State = digitalRead(button3Pin); // Baca status dari pin button 3
button4State = digitalRead(button4Pin); // Baca status dari pin button 4
button5State = digitalRead(button5Pin); // Baca status dari pin button 5
button6State = digitalRead(button6Pin); // Baca status dari pin button 6
button7State = digitalRead(button7Pin); // Baca status dari pin button 7
button8State = digitalRead(button8Pin); // Baca status dari pin button 8
```

```
// Kotak 2
button9State = digitalRead(button9Pin); // Baca status dari pin button 9
button10State = digitalRead(button10Pin); // Baca status dari pin button 10
button11State = digitalRead(button11Pin); // Baca status dari pin button 11
button12State = digitalRead(button12Pin); // Baca status dari pin button 12
button13State = digitalRead(button13Pin); // Baca status dari pin button 13
button14State = digitalRead(button14Pin); // Baca status dari pin button 14
// Kotak 3
button15State = digitalRead(button15Pin); // Baca status dari pin button 15
button16State = digitalRead(button16Pin); // Baca status dari pin button 16
button17State = digitalRead(button17Pin); // Baca status dari pin button 17
button18State = digitalRead(button18Pin); // Baca status dari pin button 18
button19State = digitalRead(button19Pin); // Baca status dari pin button 19
button20State = digitalRead(button20Pin); // Baca status dari pin button 20
// Kotak 4
button21State = digitalRead(button21Pin); // Baca status dari pin button 21
button22State = digitalRead(button22Pin); // Baca status dari pin button 22
button23State = digitalRead(button23Pin); // Baca status dari pin button 23
button24State = digitalRead(button24Pin); // Baca status dari pin button 24
button25State = digitalRead(button25Pin); // Baca status dari pin button 25
button26State = digitalRead(button26Pin); // Baca status dari pin button 26
```

```
 // Kotak 5
 button27State = digitalRead(button27Pin); // Baca status dari pin button 27
 button28State = digitalRead(button28Pin); // Baca status dari pin button 28
 button29State = digitalRead(button29Pin); // Baca status dari pin button 29
 button30State = digitalRead(button30Pin); // Baca status dari pin button 30
 button31State = digitalRead(button31Pin); // Baca status dari pin button 31
 button32State = digitalRead(button32Pin); // Baca status dari pin button 32

 // Kotak 1
////////////////////////////////////////////////////////////////////////////////////////////////////masuk
  if (buttonState == HIGH){     // Sensor Untuk Aktifin Conveyor
                                // JIka tombol button = HIGH maka,
    digitalWrite(ledPin, HIGH); // Mengatur ledPin Agar Motor Conveyor ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////
if (button2State == HIGH && button7State == HIGH){
          // Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
          // Jika tombol button 2 = HIGH dan tombol button 7 = HIGH maka,
    digitalWrite(ledPin,LOW);      // Mengatur ledPin Agar Motor Conveyor OFF
dan
    digitalWrite(led2Pin,HIGH);  // Mengatur led2Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button3State == HIGH){    // Saklar Untuk Berhentiin Motor Tarik Helm
```

```
                                    // Jika tombol button 3 = HIGH maka,
    digitalWrite(led2Pin,LOW); // Mengatur led2Pin Agar Motor Tarik Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////keluar
  if (button4State == HIGH){      // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                    // Jika tombol button 4 = HIGH maka,
    digitalWrite(led3Pin,HIGH); // Mengatur led3Pin Agar Motor Dorong ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button5State == HIGH){    // Saklar Untuk Berhentiin Motor Dorong
                                    // Jika tombol button 5 = HIGH maka,
    digitalWrite(led3Pin,LOW);  // Mengatur led3Pin Agar Motor Dorong Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button8State == HIGH && button5State == HIGH){
        // Sensor Benda dan Saklar Aktifin Motor Conveyor
        // Jika tombol button 8 = HIGH dan tombol button 5 = HIGH maka,
    digitalWrite(ledPin,HIGH); // Mengatur ledPin Agar Motor Conveyor ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button6State == HIGH){  // Sensor Berhentiin Motor Conveyor
                                    // Jika tombol button 6 = HIGH maka,
```

```
    digitalWrite(ledPin,LOW); // Mengatur ledPin Agar Motor Conveyor OFF
  }

 // Kotak 2
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button9State == HIGH && button13State == HIGH){
// Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
// Jika tombol button 9 = HIGH dan tombol button 13 = HIGH maka,
    digitalWrite(ledPin,LOW);     // Mengatur ledPin Agar Motor Conveyor OFF
    digitalWrite(led4Pin,HIGH); // Mengatur led4Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button10State == HIGH){   // Saklar Untuk Berhentiin Motor Tarik Helm
                                // Jika tombol button 10 = HIGH maka,
    digitalWrite(led4Pin,LOW); // Mengatur led4Pin Agar Motor Tarik Helm OFF
  }
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////keluar
  if (button11State == HIGH){     // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                  // Jika tombol button 11 = HIGH maka,
    digitalWrite(led5Pin,HIGH); // Mengatur led5Pin Agar Motor Dorong ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button12State == HIGH){    // Saklar Untuk Berhentiin Motor Dorong
```

```
                                    // Jika tombol button 12 = HIGH maka,
    digitalWrite(led5Pin,LOW);   // Mengatur led5Pin Agar Motor Dorong Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button14State == HIGH && button12State == HIGH){
          // Sensor Benda dan Saklar Aktifin Motor Conveyor
          // Jika tombol button 14 = HIGH dan tombol button 12 = HIGH maka,
    digitalWrite(ledPin,HIGH); // Mengatur ledPin Agar Motor Conveyor ON
  }

 // Kotak 3
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button15State == HIGH && button19State == HIGH){
          // Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
          // Jika tombol button 15 = HIGH dan tombol button 19 = HIGH maka,
    digitalWrite(ledPin,LOW);     // Mengatur ledPin Agar Motor Conveyor OFF
    digitalWrite(led6Pin,HIGH); // Mengatur led6Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button16State == HIGH){  // Saklar Untuk Berhentiin Motor Tarik Helm
                                    // Jika tombol button 16 = HIGH maka,
    digitalWrite(led6Pin,LOW); // Mengatur led6Pin Agar Motor Tarik Helm OFF
  }
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////keluar
  if (button17State == HIGH){     // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                  // Jika tombol button 17 = HIGH maka,
    digitalWrite(led7Pin,HIGH); // Mengatur led7Pin Agar Motor Dorong ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button18State == HIGH){    // Saklar Untuk Berhentiin Motor Dorong
                                  // Jika tombol button 18 = HIGH maka,
    digitalWrite(led7Pin,LOW);    // Mengatur led7Pin Agar Motor Dorong Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button20State == HIGH && button18State == HIGH){
        // Sensor Benda dan Saklar Aktifin Motor Conveyor
        // Jika tombol button 20 = HIGH dan tombol button 18 = HIGH maka,
    digitalWrite(ledPin,HIGH); // Mengatur ledPin Agar Motor Conveyor ON
  }


 // Kotak 4
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button21State == HIGH && button25State == HIGH){
        // Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
        // Jika tombol button 21 = HIGH dan tombol button 25 = HIGH maka,
```

```
    digitalWrite(ledPin,LOW);    // Mengatur ledPin Agar Motor Conveyor OFF
    digitalWrite(led8Pin,HIGH); // Mengatur led8Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button22State == HIGH){  // Saklar Untuk Berhentiin Motor Tarik Helm
                               // Jika tombol button 22 = HIGH maka,
    digitalWrite(led8Pin,LOW); // Mengatur led8Pin Agar Motor Tarik Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////keluar
  if (button23State == HIGH){    // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                 // Jika tombol button 23 = HIGH maka,
    digitalWrite(led9Pin,HIGH); // Mengatur led9Pin Agar Motor Dorong ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button24State == HIGH){  // Saklar Untuk Berhentiin Motor Dorong
                               // Jika tombol button 24 = HIGH maka,
    digitalWrite(led9Pin,LOW);  // Mengatur led9Pin Agar Motor Dorong Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button26State == HIGH && button24State == HIGH){
        // Sensor Benda dan Saklar Aktifin Motor Conveyor
        // Jika tombol button 26 = HIGH dan tombol button 24 = HIGH maka,
```

```
   digitalWrite(ledPin,HIGH);  // Mengatur ledPin Agar Motor Conveyor ON
  }


 // Kotak 5
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button27State == HIGH && button31State == HIGH){
          // Sensor Benda dan Sensor Suara Untuk Aktifin Motor Tarik Helm
          // Jika tombol button 27 = HIGH dan tombol button 31 = HIGH maka,
   digitalWrite(ledPin,LOW);       // Mengatur ledPin Agar Motor Conveyor OFF
   digitalWrite(led10Pin,HIGH); // Mengatur led10Pin Agar Motor Tarik Helm ON
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  if (button28State == HIGH){   // Saklar Untuk Berhentiin Motor Tarik Helm
                                // Jika tombol button 28 = HIGH maka,
   digitalWrite(led10Pin,LOW); // Mengatur led10Pin Agar Motor Tarik Helm OFF
  }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////keluar
  if (button29State == HIGH){      // Sensor Suara Untuk Aktifin Motor Dorong Helm
                                   // Jika tombol button 29 = HIGH maka,
   digitalWrite(led11Pin,HIGH); // Mengatur led11Pin Agar Motor Dorong ON
  }
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

  if (button30State == HIGH){    // Saklar Untuk Berhentiin Motor Dorong

                                 // Jika tombol button 30 = HIGH maka,

    digitalWrite(led11Pin,LOW); // Mengatur led11Pin Agar Motor Dorong Helm OFF

  }

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

  if (button32State == HIGH && button30State == HIGH){

        // Sensor Benda dan Saklar Aktifin Motor Conveyor

        // Jika tombol button 32 = HIGH dan tombol button 30 = HIGH maka,

    digitalWrite(ledPin,HIGH); // Mengatur ledPin Agar Motor Conveyor ON

  }

}
```

Lampiran 3

Program EasyVR 1 Kotak

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
  // Shield Jumper on HW (for Leonardo and Due)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h"

EasyVR easyvr(port);
```

```
// Groups and Perintah Suara
enum Groups
{
  GROUP_1  = 1,
};

enum Group1
{
  G1_SATU = 0,
  G1_DUA = 1,
  G1_TIGA = 2,
  G1_EMPAT = 3,
  G1_LIMA = 4,
  G1_ENAM = 5,
  G1_TUJUH = 6,
  G1_DELAPAN = 7,
  G1_SEMBILAN = 8,
  G1_SEPULUH = 9,
};

// Nomor Pada Pin Output
int8_t group, idx;
int led2 = 2; // Motor Dorong Helm
```

```
int led3 = 3; // Motor Tarik Helm
int led4 = 4; // Motor Dorong Helm
int led5 = 5; // Motor Tarik Helm
int led6 = 6; // Motor Dorong Helm
int led7 = 7; // Motor Tarik Helm
int led10 = 10; // Motor Dorong Helm
int led11 = 11; // Motor Tarik Helm
int ledA4 = A4; // Motor Dorong Helm
int ledA5 = A5; // Motor Tarik Helm

void setup() // Semua kode didalam kurung kurawal akan dijalankan hanya satu
             // kali ketika program Arduino dijalankan untuk pertama kalinya
{
  pinMode(led2,OUTPUT); // Mengatur digital pin led2 sebagai output
  pinMode(led3,OUTPUT); // Mengatur digital pin led3 sebagai output
  pinMode(led4,OUTPUT); // Mengatur digital pin led4 sebagai output
  pinMode(led5,OUTPUT); // Mengatur digital pin led5 sebagai output
  pinMode(led6,OUTPUT); // Mengatur digital pin led6 sebagai output
  pinMode(led7,OUTPUT); // Mengatur digital pin led7 sebagai output
  pinMode(led10,OUTPUT); // Mengatur digital pin led10 sebagai output
  pinMode(led11,OUTPUT); // Mengatur digital pin led11 sebagai output
  pinMode(ledA4,OUTPUT); // Mengatur digital pin ledA4 sebagai output
  pinMode(ledA5,OUTPUT); // Mengatur digital pin ledA5 sebagai output
```

```
  pcSerial.begin(9600); // Membuka serial port pc,
                        // mensetting kecepatan transmisi data ke 9600 bps
 port.begin(9600);
 group = GROUP_1; // Memulai Group
}

void action();
void loop()
{
 Serial.print("SILAHKAN BERBICARA");
       // Untuk menampilkan "SILAHKAN BERBICARA" ke serial monitor
 Serial.println(); // Menampilkan kembali data yang lain
 easyvr.recognizeCommand(group);
 while (!easyvr.hasFinished());
 idx = easyvr.getCommand();
 if (idx >= 0)
 {
  uint8_t train = 0;
  char name[32];
  if (easyvr.dumpCommand(group, idx, name, train))
  Serial.print("AKTIF "); // Untuk menampilkan "AKTIF " ke serial monitor

  Serial.println(name);
```

```
        // Untuk menampilkan kata yang di sebutkan (misal Lima atau Enam atau
           lain-lain)
  }
  else
  {
    Serial.print("SILAHKAN COBA LAGI ");
        // Untuk menampilkan "SILAHKAN COBA LAGI " ke serial monitor
    Serial.println();
  }
}

void action()
{
  switch (idx)
  {
  case G1_SATU:
     digitalWrite(led2,HIGH); // Mengatur led2 agar Motor Dorong Helm ON
     delay(3000); // Tunggu tiga detik
     digitalWrite(led2,LOW); // Mengatur led2 agar Motor Dorong Helm OFF
     break; // Berhenti
  case G1_DUA:
     digitalWrite(led4,HIGH); // Mengatur led4 agar Motor Dorong Helm ON
     delay(3000); // Tunggu tiga detik
```

```
    digitalWrite(led4,LOW); // Mengatur led4 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_TIGA:
    digitalWrite(led6,HIGH); // Mengatur led6 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led6,LOW); // Mengatur led6 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_EMPAT:
    digitalWrite(led10,HIGH); // Mengatur led10 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led10,LOW); // Mengatur led10 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_LIMA:
    digitalWrite(ledA4,HIGH); // Mengatur ledA4 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(ledA4,LOW); // Mengatur ledA4 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_ENAM:
    digitalWrite(led3,HIGH); // Mengatur led3 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led3,LOW); // Mengatur led3 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_TUJUH:
```

```
    digitalWrite(led5,HIGH); // Mengatur led5 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led5,LOW); // Mengatur led5 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_DELAPAN:
    digitalWrite(led7,HIGH); // Mengatur led7 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led7,LOW); // Mengatur led7 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_SEMBILAN:
    digitalWrite(led11,HIGH); // Mengatur led11 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(led11,LOW); // Mengatur led11 agar Motor Dorong Helm OFF
    break; // Berhenti
  case G1_SEPULUH:
    digitalWrite(ledA5,HIGH); // Mengatur ledA5 agar Motor Dorong Helm ON
    delay(3000); // Tunggu tiga detik
    digitalWrite(ledA5,LOW); // Mengatur ledA5 agar Motor Dorong Helm OFF
    break; // Berhenti
  }
}
```

Lampiran 4

ARDUINO MEGA

5V, 8-bit, 16 MHz, AVR

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

Spesifikasi

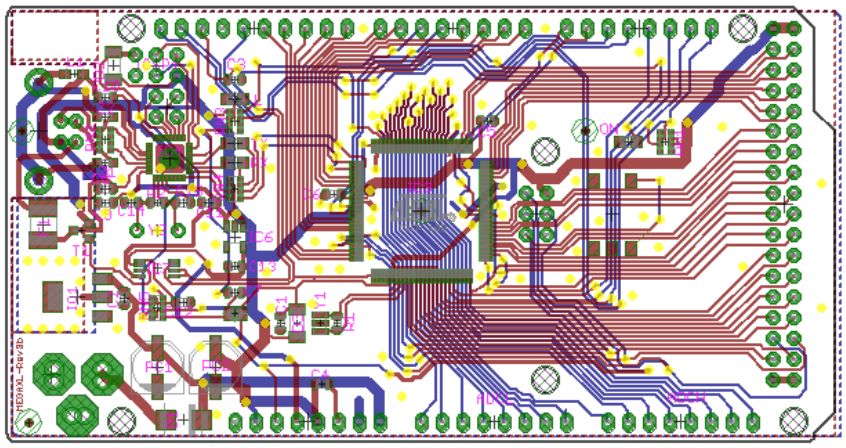
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA

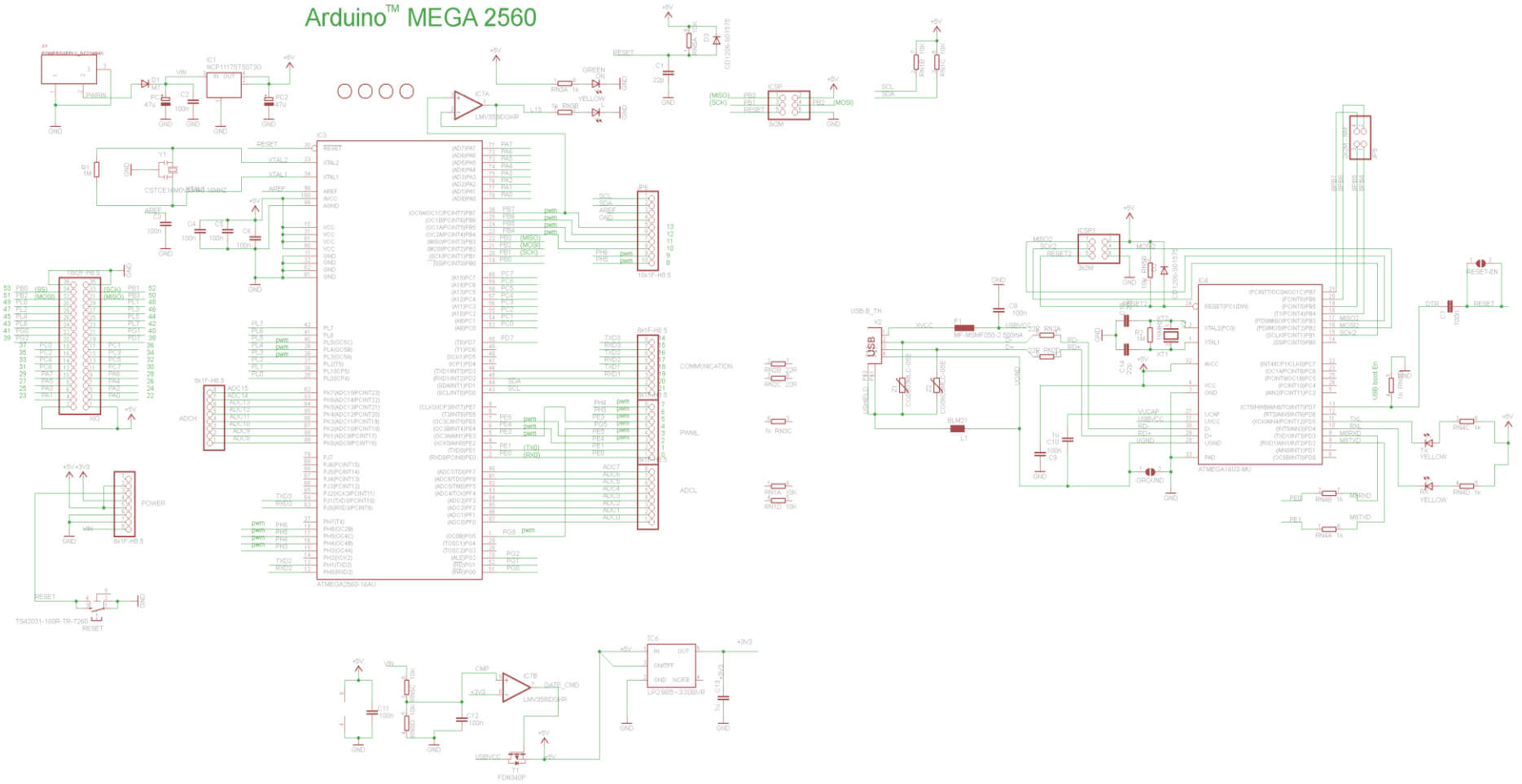
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Documentation

OSH: Schematics, Reference Design, Board size

Arduino / Genuino Mega 2560 is open-source hardware! You can build your own board using the follwing files:




Arduino™ MEGA 2560
ATMEGA2560-16AU
ATMEGA16U2-MU
COMMUNICATION
PWML
ADCL
POWER
USB
USB boot En
RESET-EN
LMV358IDGKR
LP2985-33DBVR
FDN340P
MF-MSMF050-2 500mA
CSTCE16M0V53-R0
TS42031-160R-TR-7260
RESET
GROUND

Programming

The Mega 2560 board can be programmed with the Arduino Software (IDE). For details, see the reference and tutorials.

The ATmega2560 on the Mega 2560 comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Warnings

The Mega 2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Power

The Mega 2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

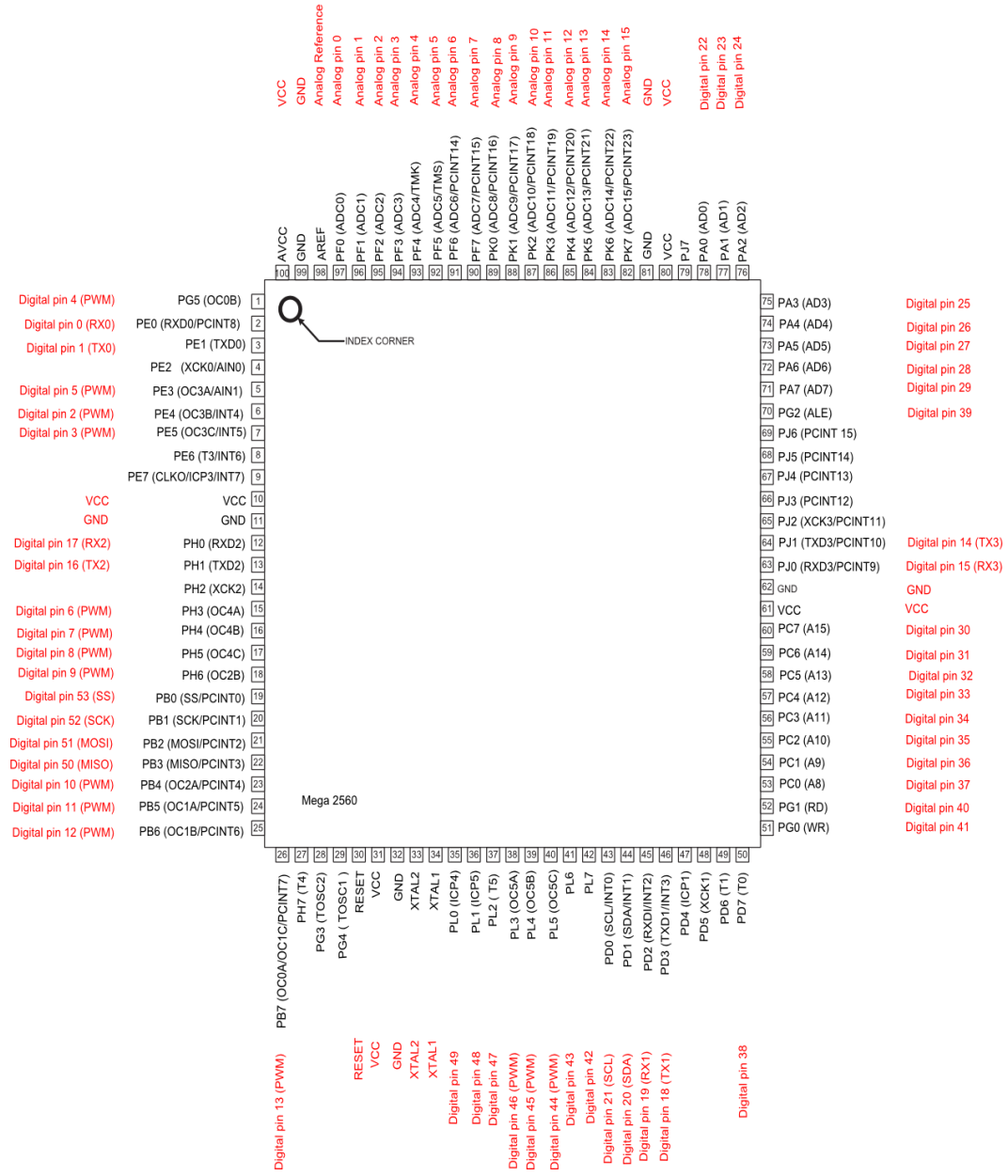
Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

See the mapping between Arduino pins and Atmega2560 ports:

Arduino Mega 2560 PIN diagram



Arduino Mega 2560 PIN mapping table

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)

3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)

23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42

43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND

63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15

83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts.

Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the attachInterrupt() function for details.
- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino /Genuino Uno and the old Duemilanove and Diecimila Arduino boards.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.

See also the mapping Arduino Mega 2560 PIN diagram.

The Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Mega 2560 board has a number of facilities for communicating with a computer, another board, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega 2560's digital pins.

The Mega 2560 also supports TWI and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the TWI bus; see the documentation for details. For SPI communication, use the SPI library.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega 2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega 2560 is designed to be compatible with most shields designed for the Uno and the older Diecimila or Duemilanove Arduino boards. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Furthermore, the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega 2560 and Duemilanove / Diecimila boards. Please note that I2C is not located on the same pins on the Mega 2560 board (20 and 21) as the Duemilanove / Diecimila boards (analog inputs 4 and 5).

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Mega 2560 is designed in a way that allows it to be reset by software running

on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega 2560 board is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the ATMega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega 2560 board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

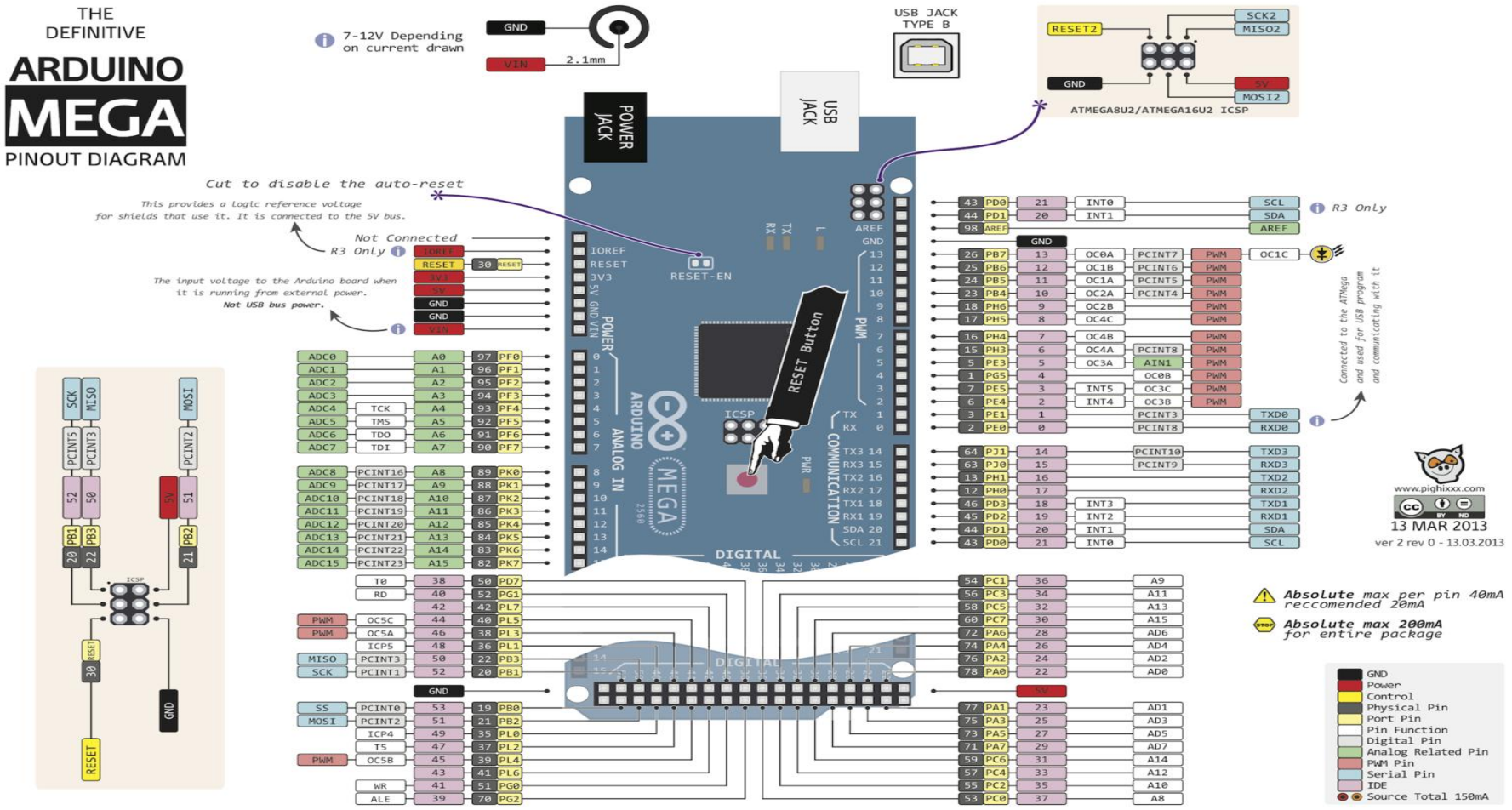
Revisions

The Mega 2560 does not use the FTDI USB-to-serial driver chip used in past designs. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and

revision 2 Arduino boards) programmed as a USB-to-serial converter. Revision 2 of the Mega 2560 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the Arduino board and the current Genuino Mega 2560 have the following improved features:

- 1.0 pinout: SDA and SCL pins - near to the AREF pin - and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the board that uses ATSAM3X8E, that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.


THE DEFINITIVE
ARDUINO MEGA
PINOUT DIAGRAM
7-12V Depending on current drawn
GND
VIN
2.1mm
USB JACK TYPE B
POWER JACK
USB JACK
RESET2
SCK2
MISO2
GND
5V
MOSI2
ATMEGA8U2/ATMEGA16U2 ICSP
Cut to disable the auto-reset
This provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power.
Not USB bus power.
RESET-EN
RESET Button
ICSP
ARDUINO MEGA
POWER
ANALOG IN
PWM
COMMUNICATION
DIGITAL
R3 Only
Connected to the ATMega and used for USB program and communicating with it
www.pighixxx.com
13 MAR 2013
ver 2 rev 0 - 13.03.2013
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

Lampiran 5

ARDUINO UNO

5V, 8-bit, 16 MHz, AVR

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Spesifikasi

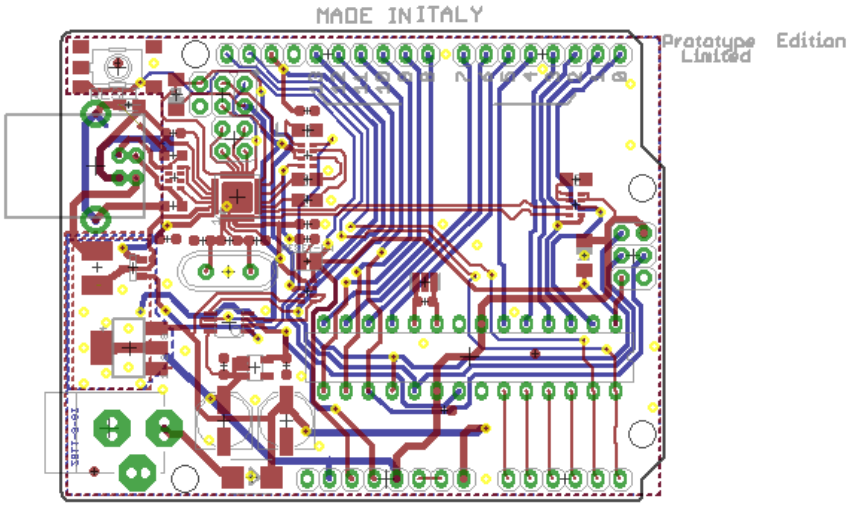
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V

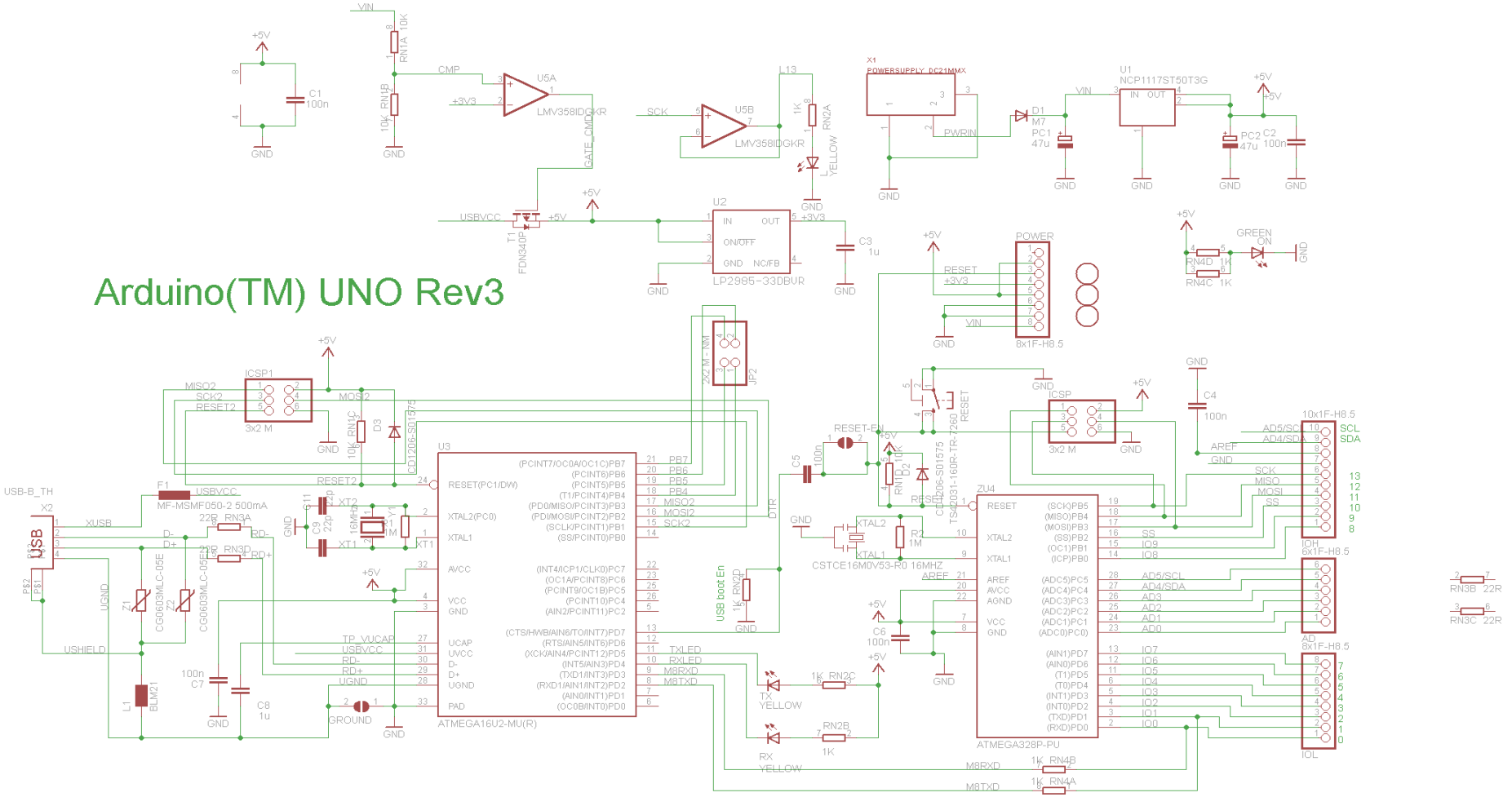
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Documentation

OSH: Schematics, Reference Design, Board size

Arduino / Genuino Uno is open-source hardware! You can build your own board using the follwing files:




Arduino(TM) UNO Rev3
ATMEGA16U2-MU(R)
ATMEGA328P-PU
LP2985-33DBVR
LMV358IDGKR
NCP1117ST50T3G
CSTCE16M0V53-R0 16MHZ
USB-B_TH
ICSP1
ICSP
POWER
10x1F-H8.5
8x1F-H8.5
6x1F-H8.5
GREEN ON
RESET-EN
USB boot En
TX YELLOW
RX YELLOW
L YELLOW
M8RXD
M8TXD
TXLED
RXLED
SCL
SDA
AD5/SCL
AD4/SDA
AD3
AD2
AD1
AD0
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
SCK
MISO
MOSI
SS
IOH
IOL
AD
GND
+5V
+3V3
VIN
USBVCC
RESET
PWRIN
XTAL1
XTAL2
AREF
USHIELD
UGND

Programming

The Arduino/Genuino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino/Genuino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then rese ing the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Warnings

The Arduino/Genuino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Differences with other boards

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Power

The Arduino/Genuino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

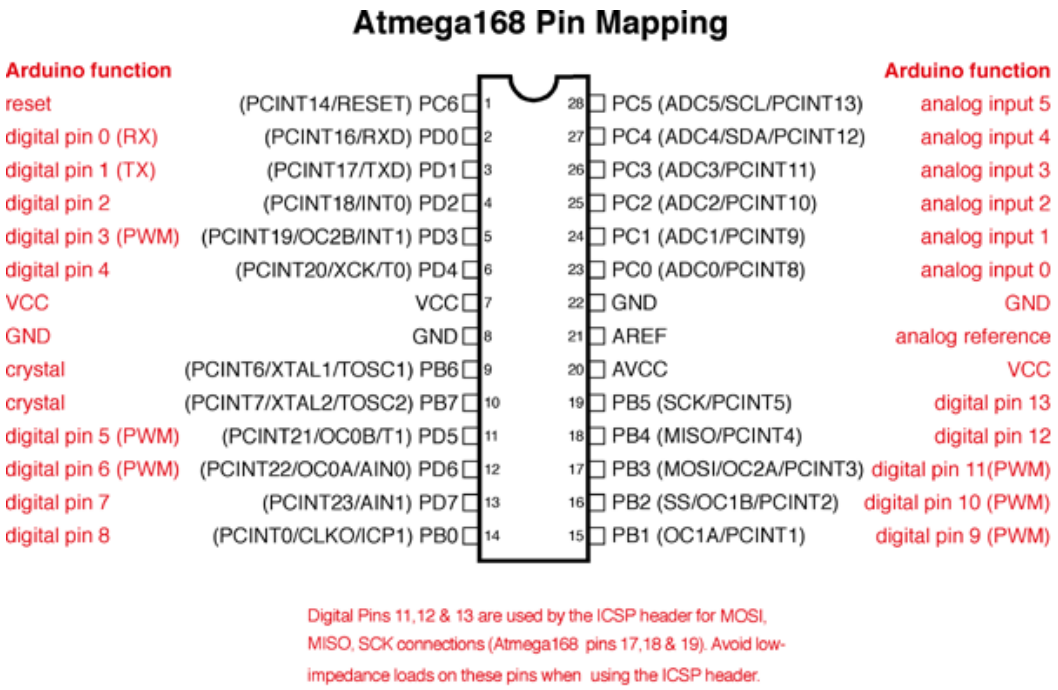
- Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

See the mapping between Arduino pins and ATmega328P ports. The mapping for the Atmega8, 168, and 328 is identical.



Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The

ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

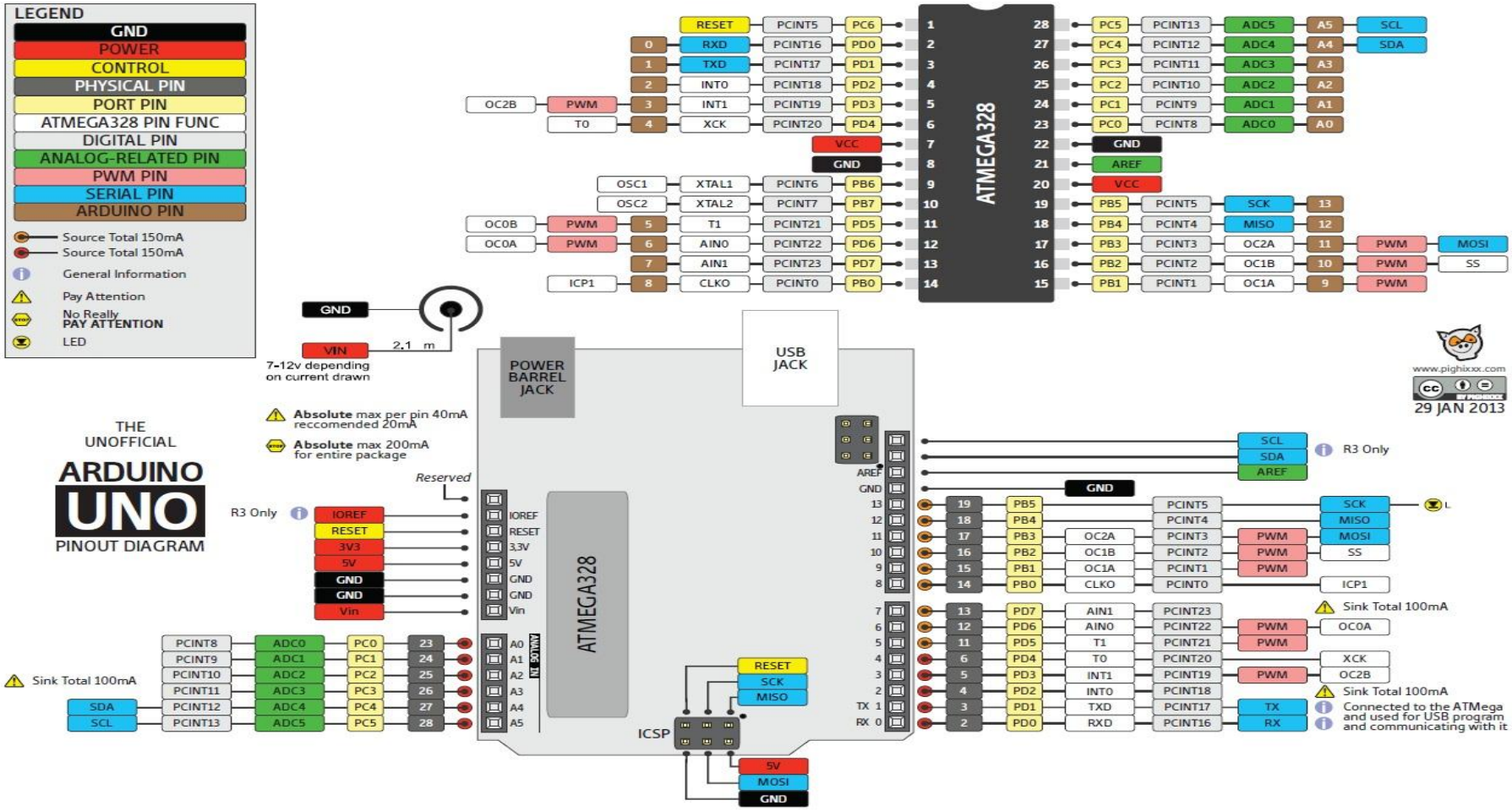
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

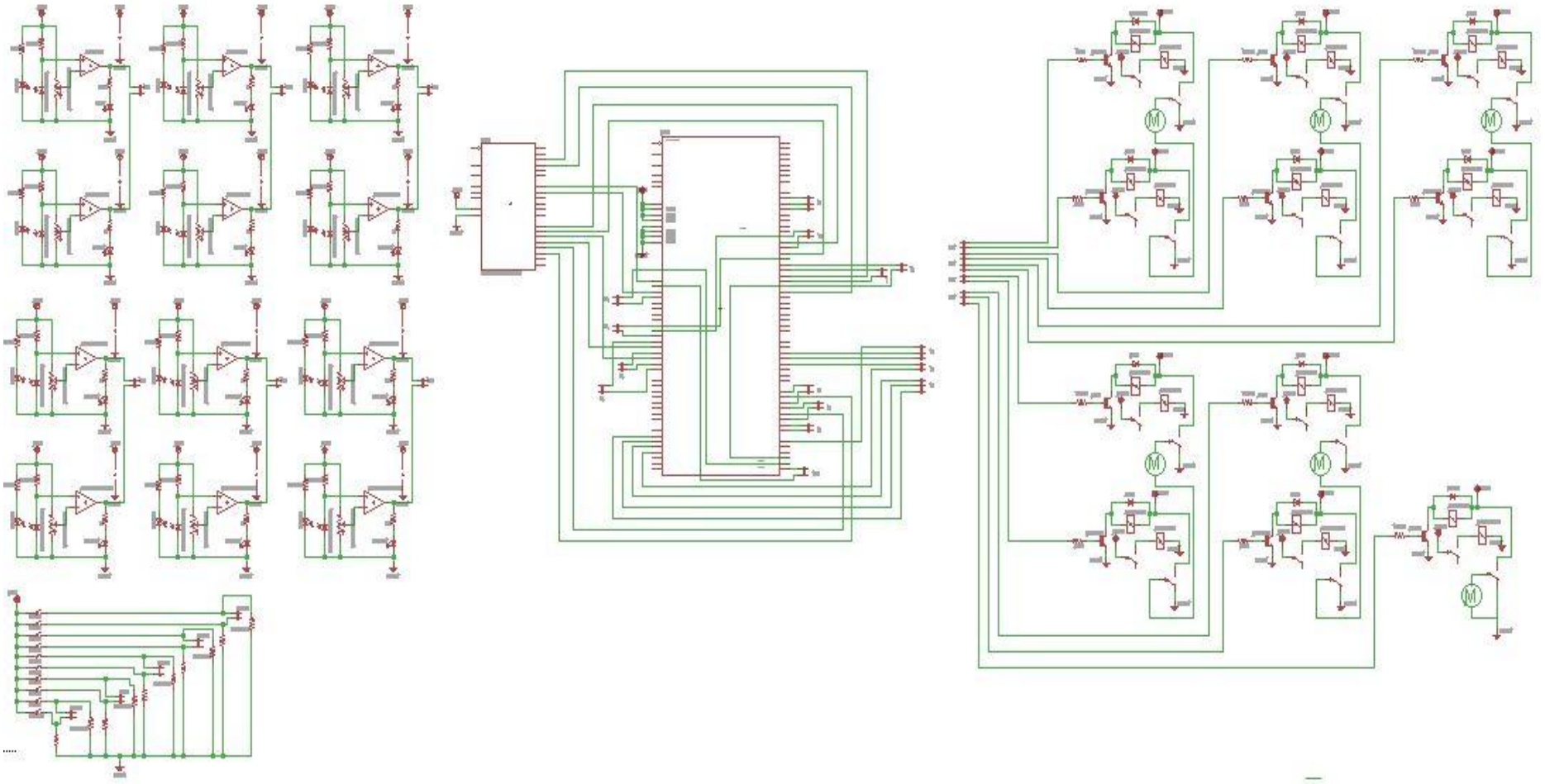
The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

Revisions

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.


LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
ATMEGA328 PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
ARDUINO PIN
Source Total 150mA
Source Total 150mA
General Information
Pay Attention
No Really
PAY ATTENTION
LED
ATMEGA328
RESET PCINT5 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
28 PC5 PCINT13 ADC5 A5 SCL
27 PC4 PCINT12 ADC4 A4 SDA
26 PC3 PCINT11 ADC3 A3
25 PC2 PCINT10 ADC2 A2
24 PC1 PCINT9 ADC1 A1
23 PC0 PCINT8 ADC0 A0
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 SCK 13
18 PB4 PCINT4 MISO 12
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
GND
VIN
2.1 m
7-12v depending on current drawn
POWER BARREL JACK
USB JACK
www.pighixxx.com
29 JAN 2013
THE UNOFFICIAL
ARDUINO UNO
PINOUT DIAGRAM
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Reserved
R3 Only
IOREF
RESET
3V3
5V
GND
GND
Vin
ATMEGA328
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
PWM
SCL
SDA
AREF
R3 Only
GND
13 PB5 PCINT5 SCK MISO MOSI SS
12 PB4 PCINT4
11 PB3 OC2A PCINT3 PWM
10 PB2 OC1B PCINT2 PWM
9 PB1 OC1A PCINT1 PWM
8 PB0 CLKO PCINT0 ICP1
Sink Total 100mA
7 PD7 AIN1 PCINT23
6 PD6 AIN0 PCINT22 PWM OC0A
5 PD5 T1 PCINT21 PWM
4 PD4 T0 PCINT20 XCK
3 PD3 INT1 PCINT19 PWM OC2B
2 PD2 INT0 PCINT18
TX 1 PD1 TXD PCINT17 TX
RX 0 PD0 RXD PCINT16 RX
Sink Total 100mA
Connected to the ATMega and used for USB program and communicating with it
Sink Total 100mA
PCINT8 ADC0 PC0 23
PCINT9 ADC1 PC1 24
PCINT10 ADC2 PC2 25
PCINT11 ADC3 PC3 26
PCINT12 ADC4 PC4 27
PCINT13 ADC5 PC5 28
SDA
SCL
ICSP
RESET
SCK
MISO
5V
MOSI
GND



111

Lampiran 6

EasyVR

VeeaR

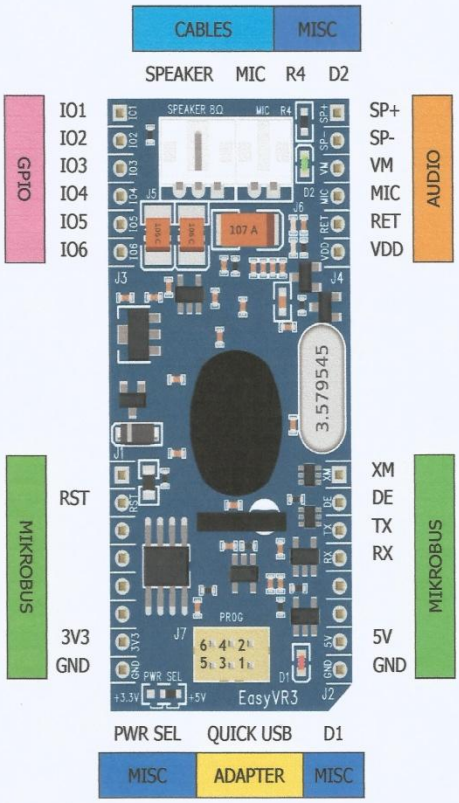
EasyVR 3

User Manual

Release 1.0.15

www.veear.eu

Technical specifications



The outer headers J1 and J2 are the mikroBUS™ interface connectors, providing selectable 3.3V/5V power input to the module and voltage translated digital I/O lines, including: UART receive/transmit lines and control pins.

The header J3 provides configurable I/O expansion lines (inputs with weak internal pull-up by default), powered at the internal logic voltage VDD.

The header J4 contains the main analog signals, such as microphone signals and amplified DAC outputs, which are also available on the internal right angle connectors J5 and J6.

The module can also be operated through the programming connector J7 alone, by using the *QuickUSB Adapter Cable*.

Pin assignment

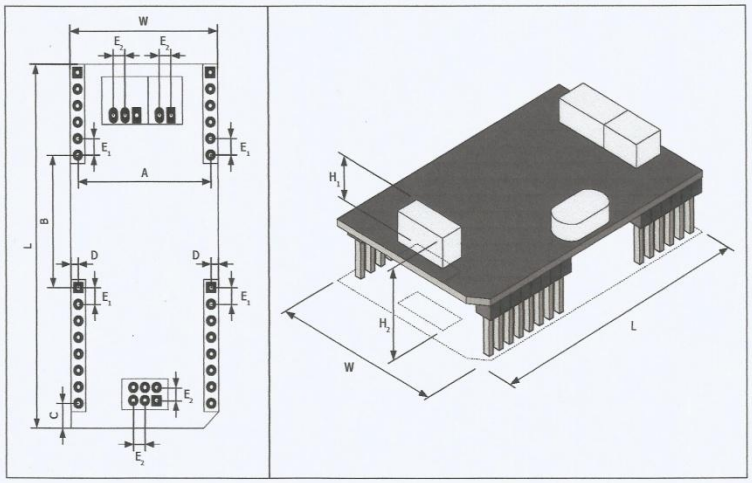
Group	Name	Number	Pin	Type	Description
MIKROBUS	J1	1	-	-	(Not connected)
		2	RST	I	Active low asynchronous reset (internal pull-up)
		3-6	-	-	(Not connected)
		7	3V3	I	3.3V DC power input
		8	GND	-	Ground
	J2	1	XM	I	Boot select (internal pull-down)
		2	DE	O	(Reserved)
		3	TX	O	Serial Data Transmit
		4	RX	I	Serial Data Receive
		5-6	-	-	(Not connected)
		7	5V	I	5.0V DC power input
		8	GND	-	Ground
GPIO	J3	1	IO1	I/O	General purpose I/O (VDD logic levels)
		2	IO2	I/O	General purpose I/O (VDD logic levels)
		3	IO3	I/O	General purpose I/O (VDD logic levels)
		4	IO4	I/O	General purpose I/O (VDD logic levels)
		5	IO5	I/O	General purpose I/O (VDD logic levels)
		6	IO6	I/O	General purpose I/O (VDD logic levels)
AUDIO	J4	1	SP+	O	Differential audio output (can directly drive 8Ω speaker)
		2	SP-	O	
		3	VM	O	Microphone power (to support custom microphones)
		4	MIC	I	Microphone audio input
		5	RET	-	Microphone return (analog ground)
		6	VDD	O	Internal logic voltage (for reference only)
CABLES	J5	1	SP-	O	Differential audio output (can directly drive 8Ω speaker)
		3	SP+	O	
		2	-	-	(Not connected)
	J6	1	MIC	I	Microphone audio input
		2	RET	-	Microphone return (analog ground)
ADAPTER	J7	1	RX_P	O	Programming cable serial data receive
		2	RTS_P	I	Programming cable request to send (reset/boot control)
		3	GND	-	Programming cable ground
		4	5V_P	I	Programming cable 5V DC power output
		5	TX_P	I	Programming cable serial data transmit
		6	CTS_P	O	Programming cable clear to send (tied to ground)

Note: The General Purpose I/O lines (J3.1-6) are at nominal 3.0VDC level. Do not connect higher voltages directly to these pins!

Settings and indicators

Group	Name	Type	Description
MISC	PWR SEL	3-Way Jumper (SMD 0603)	Select power input and voltage level between +3.3V and +5V with a zero Ohm resistor or solder bridge
	D1	LED	Red light indicator, normally ON when the board is powered, briefly blinking on serial data received
	D2	LED	Green light indicator, turns ON when the module is listening to its audio input
	R4	Resistor (SMD 0603)	Microphone gain resistor, default is 1.2kΩ

Physical dimensions



Symbol	Parameter	Units (mm / Inches)	
W	Width	25.4	1.000
L	Length	56.4	2.220
H_1	Height (without outer strips J1-J4)	9.5	0.375
H_2	Height (with outer strips J1-J4)	17.0	0.670
E_1	Connector pitch and pin spacing (of outer strips J1-J4)	2.54	0.100
E_2	Connector pitch (of inner connectors J5-J7)	2.00	0.079
A	Headers horizontal spacing	22.86	0.900
B	Headers vertical spacing	20.32	0.800
C	Header vertical offset	3.81	0.150
D	Header horizontal offset	1.27	0.050

These are applicable to pin RX_P.

Symbol	Parameter	Min	Typ	Max	Unit
V_{OH}	Output High Voltage (I_{OH} = -5 mA)	2.4		3.0	V
V_{OL}	Output Low Voltage (I_{OL} = 8 mA)	0.0		0.6	V

These are applicable to pins IO1 - IO6.

Symbol	Parameter		Min	Typ	Max	Unit
V_{IH}	Input High Voltage		2.4	3.0	3.3	V
V_{IL}	Input Low Voltage		-0.1	0.0	0.75	V
I_{IL}	Input Leakage Current (0 < V_I < 3V, Hi-Z Input)			<1	10	µA
R_{PU}	Pull-up Resistance	Strong		10		kΩ
		Weak		200		kΩ
V_{OH}	Output High Voltage (I_{OH} = -5 mA)		2.4		3.0	V
V_{OL}	Output Low Voltage (I_{OL} = 8 mA)		0.0		0.6	V

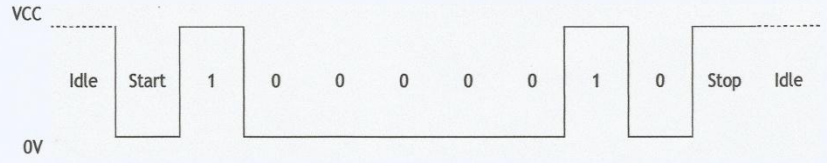
Serial Interface

The EasyVR 3 communicates via an asynchronous serial interface (commonly known as UART interface), with the following features:

- Baud Rate: **9600** (default), 19200, 38700, 57600, 115200
- Frame: **8** Data bits, **No** parity, **1** Stop bit

The receiver input data line is RX, while the transmitter output data line is TX. No handshake lines are used.

Example of a serial data frame representing character "A" (decimal 65 or hexadecimal 41):



See also chapter **Communication Protocol** later on this manual for communication details.

Microphone

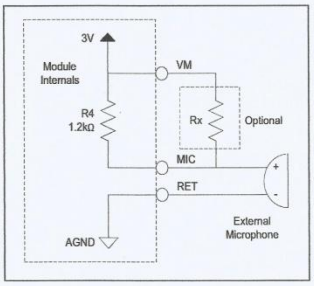
The microphone provided with the EasyVR 3 module is an omnidirectional electret condenser microphone (Horn EM9745P-382):

- Sensitivity -38dB (0dB=1V/Pa @1KHz)
- Load Impedance 2.2K
- Operating Voltage 3V
- Almost flat frequency response in the range 100Hz - 20kHz

The microphone circuit is optimized for use at ARMS_LENGTH (default, about 60cm) or FAR_MIC distance settings.

If you use a microphone with different specifications the recognition accuracy may be adversely affected. Differences in rated load impedance and sensitivity can be compensated to a certain extent by changing the microphone gain. This can be done in several ways:

- Replacing the internal gain resistor R4 (1.2kΩ)
- Adding an external resistor Rx going in parallel with R4 (it can only reduce gain, useful for HEADSET distance settings)
- Removing the internal resistor R4 and using only the external resistor Rx



Microphone circuit

Modifying gain resistance

You can calculate the overall microphone gain resistance using the formula below:

$$R_s = I \times 10^{\frac{G-S}{20}}$$

Rs is the optimal microphone gain resistance
I is the impedance rating of the microphone
G is the desired overall system gain, defined as follows:

1. If the module is configured for HEADSET microphone distance (typically a few centimeters from the user's mouth), then the overall system gain should be -49 dB (0dB=1v/Pa@1KHz);
2. If the module is configured for ARMS_LENGTH microphone distance (typically 60-90 cm from the user's mouth - this is the default setting of EasyVR), then the overall system gain should be -44 dB;
3. If the module is configured for FAR_MIC microphone distance (up to about 3 meters from the user's mouth), then the overall system gain should be -43 dB.

S is the sensitivity rating of the microphone you want to use, and it is specified in -dB in the microphone's specification[3].

[3] Converting µBars to Pascal: microphone manufacturers specify the sensitivity referencing to µBars or Pascal. If the microphone sensitivity is referenced to µBars, simply add 20 dB to the rating. For example, -58 dB/µBars + 20dB = -38 dBV/Pa.

Quick start guide for using the module

Assembly notes

The EasyVR 3 is provided with separate standard 2.54mm-pitch male headers that can be used to connect the module to a breadboard, prototyping board, custom boards or carrier boards like the EasyVR Shield 3.

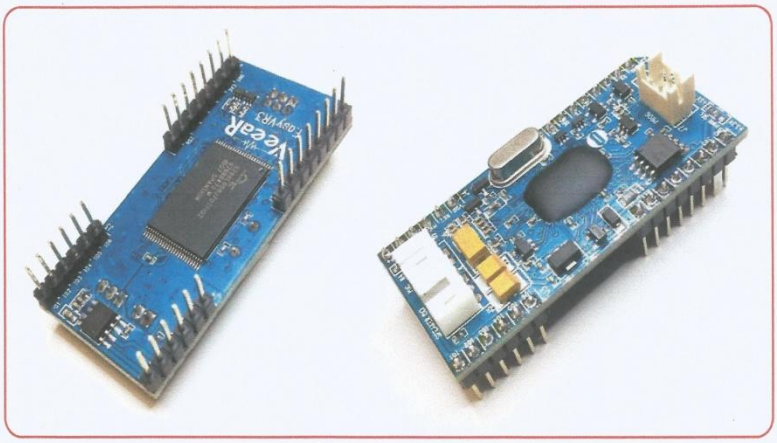
When male headers are necessary, make sure they are well soldered on the module in order to prevent electrical issues.

Some practical guides to hand soldering:

- Adafruit Guide To Excellent Soldering
- Sparkfun How to Solder: Through-Hole Soldering

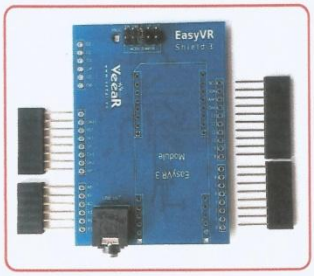
When assembling the module, especially in preparation for the EasyVR Shield 3, make sure you insert male headers from the bottom side and solder them on the top side, and that all the headers are straight and vertical for a smooth insertion on the carrier board.

The end result should be similar to the following pictures:



EasyVR Shield 3 for Arduino

Product description



The EasyVR Shield 3 is an adapter board for the EasyVR 3 module, designed to simplify its use among the Arduino community.

The Shield is compatible with any Arduino board using UNO-R3 Shield headers, running at either 3.3V or 5V levels, by using the IOREF pin to select the EasyVR operating voltage.

It is also backward compatible with earlier Arduino boards that don't have the IOREF pin, which are using 5V I/O levels by default.

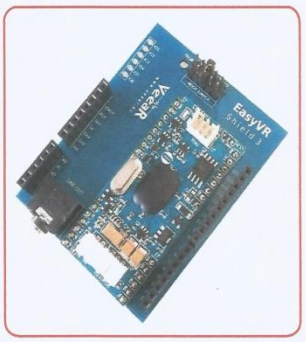
If your board does not have the IOREF pin but it is running at 3.3V, you can still operate the EasyVR Shield 3 correctly if you manually connect pins IOREF and 3V3 together, for example with a jumper wire.

The board comes with separate Arduino stackable headers for the Shield interface. The EasyVR 3 module is also provided separately.

Note: The EasyVR 3 module and all stackable headers must be soldered before use!

EasyVR Shield 3 Features

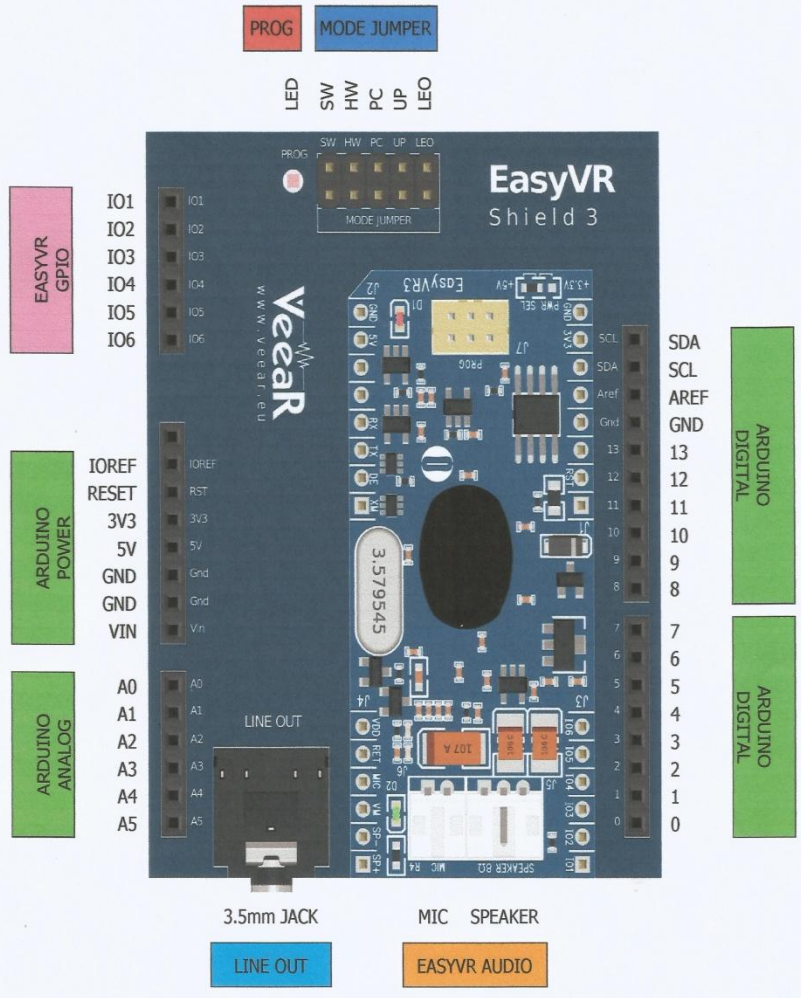
- Compatible with Arduino boards that have the 1.0 Shield interface (UNO R3) including, but not limited to:
  - Arduino Zero
  - Arduino Uno
  - Arduino Mega
  - Arduino Leonardo
  - Arduino Due
- Supports 5V and 3.3V main boards through the IOREF pin (defaults to 5V if this pin is absent)
- Supports direct connection to the PC on main boards with a separate USB/Serial chip and a special software-driven "bridge mode" on boards with only native USB interface, for easy access and configuration with the EasyVR Commander
- Enables different modes of serial connection and also flash updates to the embedded EasyVR module (through the Mode Jumper)
- Supports remapping of serial pins used by the Shield (in SW mode)
- Provides a 3.5mm audio output jack suitable for headphones or as a line out



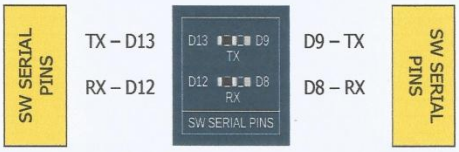
EasyVR Shield 3 fully assembled

Technical specifications

Board overview



(Top View)



(Detail - Bottom View)

Pin assignment

Group	Pin	Description
ARDUINO HEADERS	-	Arduino UNO-R3 Shield interface, pass-through connectors (Pins 0-1 are in use when J12 is set to UP, PC, HW or LEO) (Pins 12-13 or 8-9 are in use when J12 is set to SW)
EASYVR AUDIO	-	Audio cables connectors of the EasyVR 3 module (microphone and speaker)
LINE OUT	-	3.5mm stereo/mono jack (16Ω - 32Ω headphones or line-level output)
MODE JUMPER	SW	Arduino Software Serial (connected to pins 12-13 or 8-9)
	HW	Arduino Hardware Serial (connected to pins 0-1)
	PC	PC Mode (Arduino disabled, EasyVR in command mode)
	UP	Update Mode (Arduino disabled, EasyVR in boot mode)
	LEO	Leonardo Update (Arduino enabled, EasyVR in boot mode)
PROG	-	Red light indicator for Flash programming modes (UP and LEO)
SW SERIAL PINS	RX	Use resistor to select Software Serial RX pin: 12 or 8
	TX	Use resistor to select Software Serial TX pin: 13 or 9
EASYVR GPIO	IO1	General purpose I/O as found on the embedded EasyVR 3 module (referenced at the internal VDD logic level - see note below)
	IO2	
	IO3	
	IO4	
	IO5	
	IO6	

Note: The General Purpose I/O lines (IO1-IO6) are at nominal 3.0VDC level. Do not connect higher voltages directly to these pins!

Mode Jumper settings

This jumper selects the operating mode of the EasyVR Shield and it can be placed in one of four positions:

- **SW - Software Serial mode**
  Use it for controlling the EasyVR module from your Arduino sketch through a software serial port (using pins 12-13). You can also connect the EasyVR Commander in this mode, provided that the running sketch implements bridge mode (see the Arduino library examples).
- **HW - Hardware Serial mode**
  Use it for controlling the EasyVR module from your Arduino sketch through the hardware serial port (using pins 0-1).
- **PC - PC Connection mode**
  Use it for direct connection with the EasyVR Commander. In this mode, the Arduino controller is held in reset and only the embedded USB/Serial adapter is used.
- **UP - Flash Update mode**
  Use it for firmware updates or to download sound table data and custom grammars to the on-board flash memory from the EasyVR Commander. In this mode, the Arduino controller is held in reset and only the embedded USB/Serial adapter is used. The EasyVR module is set in boot mode.
- **LEO - Leonardo Update mode**
  This is similar to the regular *Flash Update* mode, for Arduino boards that don't have a separate USB/Serial adapter, such as Arduino Leonardo. The EasyVR module is set in boot mode, but the Arduino controller is not reset and it must be running the special "bridge" sketch.

Software Serial Pins settings

On the bottom side of the board there are two SMD resistors that you can move to select the two pins of Arduino that the EasyVR will be connected to when in Software Serial mode (*Mode Jumper* on **SW**).

- **RX - Software Serial Receiver pin**
  - D12 - Use digital pin 12 as serial receiver (default)
  - D8 - Use digital pin 8 as serial receiver
- **TX - Software Serial Transmitter pin**
  - D13 - Use digital pin 13 as serial transmitter (default)
  - D9 - Use digital pin 9 as serial transmitter

The choice of pins 12-13 is maintained for backward compatibility with the previous hardware revisions of the EasyVR Shield. However those pins may also be used for the SPI interface, so another choice of pins 8-9 is provided. If you want to use different pins make sure the receiver pin supports change interrupts.