Laporan

DEBUGGING PADA ARDUINO UNO ATMEGA328

Mata Kuliah

SISTEM MIKROPROSESOR

Anggota Kelompok :

Rizky Ramadhani	04161066
Shintya Faradila	04161069

PROGRAM STUDI TEKNIK ELEKTRO

JURUSAN TEKNOLOGI INDUSTRI DAN PROSES

INSTITUT TEKNOLOGI KALIMANTAN

2018

DAFTAR ISI

BAB I
DASAR TEORI

1.1 Arduino

Arduino adalah sebuah platform komputasi fisik (*board minimum system microcontroller*) yang bersifat *open source* berbasiskan rangkaian I/O dan lingkungan pengembangan yang mengimplementasikan bahasa *processing*. Arduino dapat digunakan untuk mengembangkan obyek interaktif mandiri atau dapat dihubungkan ke perangkat lunak pada komputer anda (seperti *flash*, pengolahan, VVVV, atau Max/MSP). Rangkaiannya dapat dirakit dengan tangan atau dibeli. IDE (*Integrated Development Environment*) Arduino bersifat *open source* [1].

1.2 Arduino Uno

Arduino Uno adalah sebuah board mikrokontroler yang didasarkan pada ATmega328. Didalam rangkaian board arduino terdapat mikrokontroler AVR seri ATMega328 yang merupakan produk dari Atmel. Arduino memiliki kelebihan dibanding board microcontroller yang lain selain bersifat open source yaitu arduino mempunyai bahasa pemrogramanya sendiri yang berupa bahasa C. Selain itu dalam board arduino sendiri sudah terdapat loader yang berupa USB sehingga memudahkan kita ketika melakukan pemprograman microcontroller didalam Arduino.

Sifat open source arduino juga banyak memberikan keuntungan tersendiri untuk kita dalam menggunakan board ini, karena dengan sifat open source komponen yang kita pakai tidak hanya tergantung pada satu merek, namun memungkinkan kita bisa memakai semua komponen yang ada dipasaran [2].

Gambar 1.1 Arduino Uno

Tabel 1.1 Spesifikasi Arduino Uno

Mikrokontroler	ATmega328
Tegangan pengoperasian	5V
Tegangan input yang disarankan	7-12V
Batas tegangan input	6-20V
Jumlah pin I/O digital	14 (6 di antaranya menyediakan keluaran PWM)
Jumlah pin input analog	6

Arus DC tiap pin I/O	40 mA
Arus DC untuk pin 3.3V	50 mA
Memori Flash	32 KB (ATmega328), sekitar 0.5 KB digunakan oleh bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

1.3 Debugging

Debugging (atau pengujian program) adalah proses membuat program berperilaku sebagaimana dimaksud. Perbedaan antara perilaku yang dimaksudkan dan perilaku aktual disebabkan oleh 'bug' (kesalahan program) yang harus diperbaiki selama proses debugging.

Debug sering dianggap masalah karena tiga alasan:

1. Prosesnya terlalu merugikan (terlalu banyak usaha).
2. Setelah debugging, program masih mengalami bug.
3. Ketika program ini kemudian dimodifikasi, bug dapat muncul di tempat-tempat yang benar-benar tidak terduga.

Secara umum, ada dua sumber untuk masalah ini: desain program yang buruk dan teknik debugging yang buruk. Misalnya, masalah debugging yang terlalu merugikan mungkin karena adanya banyak bug (rancangan program yang buruk), atau pada teknik debugging di mana terlalu sedikit bug ditemukan untuk setiap uji coba atau setiap hari (teknik debugging yang buruk) [3].

1.4 USB

USB merupakan port masukan/keluaran baru yang dibuat untuk mengatai kekurangan-kekurangan port serial maupun paralel yang sudah ada. USB dibuat dengan kelebihan-kelebihan sebagai berikut [4]:

1. Hot-plugable, yang berarti piranti masukan/keluaran yang menggunakan USB dapat ditambahkan ketika PC menyala.
2. Mudah digunakan karena piranti masukan/keluaran yang terpasang dikenali oleh PC menggunakan driver yang sesuai kemudian konfigurasinya akan dikerjakan secara otomatis.
3. Semua piranti dipasang menggunakan satu tipe konektor.
4. Kecepatan USB sangat tinggi, dapat mencapai 12 Mbps yang tentunya jauh lebih cepat dibanding port serial dan paralel yang ada saat ini.
5. Jumlah piranti yang dapat dipasang pada 1 PC mencapai 127 piranti (dengan bantuan hub yang dapat dipasang sampai 5 tingkat), suatu batasan yang sangat tinggi untuk ekspansi piranti masukan/keluaran.
6. Piranti dengan USB dapat menggunakan catu dari PC (untuk penggunaan arus tidak lebih dari 500mA) sehingga tidak membutuhkan tambahan catu daya luar.
7. Hemat listrik karena piranti dapat mati secara otomatis apabila tidak digunakan (PC dalam keadaan suspend).
8. Adanya deteksi dan pemulihan kesalahan yang handal. Kesalahan data dideteksi dan transaksi diulang lagi untuk memastikan data dikirim/diterima dengan benar.

9. Merupakan piranti eksternal PC sehingga tidak perlu membuka kotak PC atau merancang suatu kartu antarmuka dalam penggunaan piranti masukan/keluaran dengan USB.

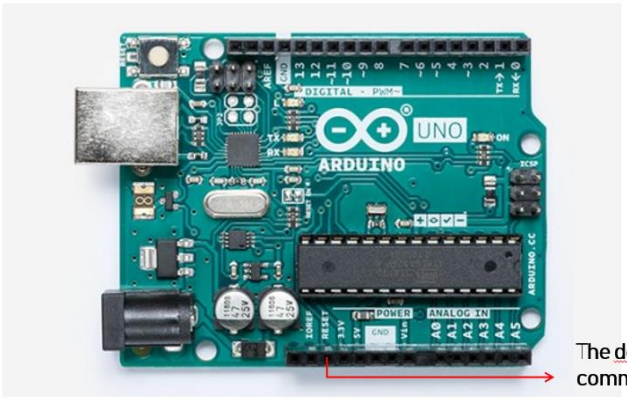
1.5 User Interface

Setiap teknologi informasi memiliki interface atau antarmuka yang berfungsi untuk menjembatani antara pengguna dengan teknologi itu sendiri. Teknologi informasi yang satu dengan yang lain memiliki desain interface yang berbeda-beda sesuai dengan fungsi dan kebutuhan penggunanya. Misalnya, teknologi yang digunakan oleh seorang dokter dengan kebutuhannya untuk memberikan diagnosa kepada pasien akan berbeda dengan teknologi informasi yang digunakan oleh seorang manajer dalam nentukan strategi pada perusahaannya. Ben Shneiderman mengatakan bahwa pada level individu, user interface dapat mengubah hidup banyak orang, misalnya user interface yang efektif untuk para profesional artinya bahwa seorang dokter dapat melakukan diagnosis lebih akurat dan pilot dapat menerbangkan pesawat lebih aman [5].

Pembuatan user interface bertujuan untuk menjadikan teknologi informasi tersebut mudah digunakan oleh pengguna atau disebut dengan istilah user friendly. Istilah user friendly digunakan untuk menunjuk kepada kemampuan yang dimiliki oleh perangkat lunak atau program aplikasi yang mudah dioperasikan, dan mempunyai sejumlah kemampuan lain sehingga pengguna merasa betah dalam mengoperasikan program tersebut [6].

BAB II
PEMBAHASAN

2.1 *Hardware*



Gambar 2.1 Tampilan *Hardware* Arduino beserta pin untuk *debugging*

Pin komunikasi debugWIRE (dW) secara fisik terletak pada pin yang sama seperti Reset Eksternal (RESET). Oleh karena itu, sumber Reset Eksternal tidak didukung ketika debugWIRE diaktifkan. Fuse DWEN yang diprogram memungkinkan beberapa bagian dari sistem jam berjalan di semua mode tidur. Ini akan meningkatkan konsumsi daya saat tidur. Dengan demikian, Sekering DWEN harus dinonaktifkan ketika debugWire tidak digunakan.

2.2 *Software*

Gambar 2.2 Software yang dapat digunakan untuk *debugging*

2.2.1 *DebugWIRE*

DebugWire adalah protokol Atmel untuk mendebug banyak ATTiny (mis. ATTiny 13, 85) dan ATmegas (mis. ATmega48 / 88/168/328) tanpa JTAG, hanya melalui pin Reset. Protokol DebugWire tidak didokumentasikan oleh Atmel tetapi beberapa orang melakukan rekayasa ulang bagian besar dari protokol, dan mampu membangun beberapa debugger sederhana. Dengan menggunakan debugWIRE seseorang memiliki akses baca dan tulis penuh ke semua memori dan kontrol penuh atas alur eksekusi. Ini mendukung satu langkah, run-to-kursor, step-out, dan instruksi pemecahan perangkat lunak.

2.2.2 Visual Micro

Visual micro adalah plugin yang tersedia melalui Microsoft Visual Studio yang digunakan untuk membuat program lintas platform di Arduino. Kode apa pun yang dibuat dalam Visual Micro yang mematuhi Arduino akan diterima. Visual Micro sangat bagus untuk tim kolaboratif yang mendebug Arduino karena memungkinkan pengeditan kode dan perpustakaan bersama. Kode dapat dibuat di berbagai platform dan digabungkan dengan kode program di seluruh proses pembuatan. Visual Micro juga menawarkan debugging GDB dan Serial, Bluetooth dan Wifi debugging.

2.2.3 Atmel Studio

Atmel Studio IDE adalah perangkat lunak gratis yang menawarkan fasilitas debugging kompetitif untuk membantu menyelesaikan kesalahan kode. Melalui Atmel Studio, proyek dapat dikembangkan, dikompilasi dan diunggah ke mikroprosesor yang relevan. Seperti Arduino IDE, IDE Atmel Studio menggunakan kode yang sama, artinya Anda tidak perlu mempelajari bahasa pemrograman baru untuk menggunakannya. Atmel sangat serbaguna untuk komunitas open source dan mendukung sketsa .ino dan kode sumber C ++. Atmel Studio juga menyediakan kemampuan debugging yang sangat baik menggunakan DebugWire atau JTAG.

2.3 Metode Pendekatan Debugging

Secara umum, metode pendekatan debugging yaitu :

1. Gaya Kasar (Brutal Force)

Gaya ini merupakan yang paling umum dan efisien untuk memisahkan antara penyebab kesalahan perangkat dengan aplikasi. Metode ini diterapkan apabila semua cara yang lain telah gagal dilakukan.

2. Penelusuran Balik (Backtracking)

Biasanya dilakukan pada aplikasi kecil yang sangat mudah untuk dilacak kesalahannya. Data yang berhubungan dengan aplikasi lebih mudah untuk dilacak dan dicari kesalahannya karena aplikasi kecil biasanya tidak membutuhkan source hardware yang banyak.

3. Eliminasi Masalah

Mengumpulkan data yang saling berkaitan diperlukan untuk mencari dimana letak kesalahan dari aplikasi yang dibuat. Membuat hipotesis dari data yang berhasil dikumpulkan. Selanjtnya melakukan pengujian berdasarkan hasil dari hipotesis yang telah dibuat. Jika hasil pengujian berhasil, maka data itu difilter dan dimasukkan sebagai salah satu metode dalam mengisolasi bug.

Masing - masing dari pendekatan debugging tersebut dapat ditambah dengan piranti debugging. Kita dapat mengaplikasikan berbagai compiler debugging yang luas, dinamis (tracer), generator test case, ruang sisa memory.

2.4 Step Untuk Debugging

Gambar 2.3 *Step* untuk *Debugging*

Jika kode uji Anda berfungsi, sekarang waktunya menulis kode Anda sendiri. Saat menulis kode Arduino Anda sendiri, ada sejumlah aturan dasar dan praktik terbaik yang harus Anda ikuti. Dengan mengikuti petunjuk di bawah ini, Anda akan memiliki waktu yang jauh lebih mudah jika Anda perlu melakukan debug nanti:

1. Tulis kode dalam potongan kecil dan uji masing-masing.
2. Berikan nama yang bermakna untuk variabel dan fungsi
3. Gunakan fungsi
4. Gunakan konstanta, bukan angka
5. Tulis komentar untuk menjelaskan pilihan pengkodean untuk referensi dimasa mendatang
6. Pastikan kode Anda memiliki lekukan yang sesuai dan tetap dapat dibaca setiap saat. Anda dapat menggunakan Alt-t untuk auto-indent seluruh sketsa.

2.4.1 Kompilasi

Jika Anda mendapatkan kesalahan kompilasi saat mencoba mengkompilasi atau mengunggah kode ke forum Anda, periksa kesalahan dalam sintaks, kesalahan ketik, dan lainnya. Menggunakan sintaks yang benar sangat penting untuk memastikan kode Anda dikompilasi. Ketika kompilasi gagal, IDE akan memberi Anda kesalahan di bagian bawahnya. Namun, pesan kesalahan yang dihasilkan oleh Arduino IDE terbatas dalam deskripsi mereka dan oleh karena itu tidak selalu sangat membantu.

Jika Anda memahami arti pesan kesalahan, coba perbaiki masalah yang ditunjukkan dan dikompilasi lagi. Ini adalah bentuk pemecahan masalah Arduino yang nyaman.

Namun, jika Anda mengatasi kesalahan yang tercantum di IDE dan sketsa masih tidak dikompilasi, itu ide yang baik untuk menggunakan Google untuk mencari solusi untuk masalah tersebut. Kesalahan sintaks yang umum adalah:

- hilang; di akhir setiap baris
- salah atau tanda kurung yang hilang {}
- salah ketik
- variabel dan fungsi yang tidak terdefinisi

Arduino memiliki komunitas open source besar dengan panduan pemecahan masalah Arduino yang luas, yang akan membantu Anda mengidentifikasi masalah dengan kode Anda. Setelah Anda memecahkan masalah dan kode Anda berfungsi, klik jalankan.

2.4.2 Menjalankan Kode

Jika kode Anda menghimpun dan berhasil diunggah ke papan, tetapi tidak berjalan seperti yang diharapkan, Anda perlu memulai proses debug.

2.4.3 Serial Monitor

Sebagaimana diuraikan di atas, jika kode Anda gagal ketika Anda mencoba untuk menjalankan sketsa Anda, Anda perlu memulai debug kode Anda. Pertama, pikirkan dan tentukan parameter mana yang akan dicetak dan gunakan monitor serial untuk memonitornya di layar (Anda dapat mempelajari dasar-dasar serial cetak di sini). Tujuannya adalah untuk mencetak ikhtisar dari keadaan saat ini dari program. Dengan demikian:

1. Variabel
2. Masukan - pembacaan sensor
3. Cetakan yang menunjukkan alur program, seperti di dalam pernyataan ‘if’ untuk melihat apakah kondisi tersebut terpenuhi
4. Keluaran - mis. Nilai PWM sebelum menuliskannya ke pin
5. Hal lain yang Anda anggap penting untuk dicetak ke layar

2.4.4 Periksa Kode Secara Manual

Satu cara bagus untuk menilai kualitas kode Anda dan memeriksa kesalahan adalah dengan memeriksa kode Anda secara manual. Anda dapat melakukan ini dengan:

1. Menulis kode Anda dari awal dan memecahnya untuk melihat apakah Anda membuat kesalahan di sepanjang jalan (Paling umum, ini akan menjadi kesalahan sintaks).
2. Meninjau kembali desain Anda dan menanyakan apa logika utama Anda. Tuliskan dengan kata-kata dan kemudian lihat apakah pernyataan kode Anda menjawab pertanyaan ini. Kadang-kadang ketika kita masuk ke dalam pemrograman kita membuat kesalahan dengan berfokus pada detail-detail kecil dan kehilangan arah keseluruhan proyek. Dalam keadaan ini, mudah kehilangan sambungan ke ide awal.
3. Periksa kode Anda dan tambahkan komentar (sebanyak yang diperlukan) untuk menjelaskan kepada diri Anda apa yang Anda coba lakukan.

BAB III
KESIMPULAN

Debugging adalah proses membuat program berperilaku sebagaimana dimaksud. Perbedaan antara perilaku yang dimaksudkan dan perilaku aktual disebabkan oleh 'bug' (kesalahan program) yang harus diperbaiki selama proses debugging. Pin komunikasi debugWire terdapat pada pin yang sama dengan Reset Eksternal (RESET). Software yang dapat digunakan untuk *debugging* antara lain debugWire, Visual Micro dan Atmel Studio. Metode Pendekatan dalam melakukan *debugging* adalah Gaya Kasar (Brutal Force), Penelusuran Balik (Backtracking) dan Eleminasi Masalah. Step untuk melakukan *debugging* yaitu kompilasi, menjalankan kode, serial monitor dan memeriksa kode secara manual

DAFTAR PUSTAKA

[1] A. Kadir. *Buku Pintar Pemograman Arduino*, MediaKom, Yogyakarta, 2014.

[2] M. Banzi. *Getting started with Arduino,* Sebastopol: O'Reilly Media,Inc., 2011.

[3] Lauesen, Soren. *Software-Practice and Experience,* University of Copenhagen, Denmark, 1979.

[4] Hyde, John. *USB design by example : a practical guide to building I/O devices*, John Wiley & Sons, Inc., Toronto, 1999.

[5] Shneiderman, B., & Plasant, C. *Designing User Interface*, Pearson Education Inc., United States of America, 2005.

[6] Santosa, I. *Interaksi Manusia dan Komputer*, Andi, Yogyakarta, 2004.