RANCANG BANGUN JAM UNTUK TUNANETRA DENGAN PENUNJUK WAKTU SHOLAT BERBASIS *MICROCONTROLLER*

Oleh:

Nama : Daniel Kristianto Haryono

NIM : 07.41020.0017

Program : S1 (Strata Satu)

Jurusan : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA DAN TEKNIK KOMPUTER

SURABAYA

2011

RANCANG BANGUN JAM UNTUK TUNANETRA DENGAN PENUNJUK WAKTU SHOLAT BERBASIS *MICROCONTROLLER*

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Komputer

Oleh:

Nama : DANIEL KRISTIANTO HARYONO

NIM : 07.41020.0017

Program : S1 (Strata Satu)

Jurusan : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA DAN TEKNIK KOMPUTER

SURABAYA

2011

It's impossible for one to tell that they

themselves have no talent

Kupersembahkan untuk:

Keluargaku yang tercinta,

para Pendidikku yang terhormat, serta

Sahabat-sahabatku yang kukasihi.

Tugas Akhir

RANCANG BANGUN JAM UNTUK TUNANETRA DENGAN PENUNJUK WAKTU SHOLAT BERBASIS *MICROCONTROLLER*

dipersiapkan dan disusun oleh

Daniel Kristianto Haryono

N.I.M : 07.41020.0017

Telah diperiksa, diuji dan disetujui oleh Dewan Penguji
pada : November 2011

Susunan Dewan Penguji

Pembimbing

I. **Harianto, S.Kom., M.Eng.** _____

II. **Madha Christian Wibowo, S.Kom.** _____

Penguji

I. **Helmy Widyantara, S.Kom., M.Eng.** _____

II. **Susijanto Tri Rasmana, S.Kom., M.T.** _____

Tugas Akhir ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Sarjana

Pantjawati Sudarmaningtyas, S.Kom, OCA.
Wakil Ketua Bidang Akademik

PERNYATAAN

Dengan ini saya menyatakan dengan benar, bahwa Tugas Akhir ini adalah asli karya saya, bukan plagiat baik sebagian maupun apalagi keseluruhan. Karya atau pendapat orang lain yang ada dalam Tugas Akhir ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya. Apabila dikemudian hari ditemukan adanya tindakan plagiat pada karya Tugas Akhir ini, maka saya bersedia untuk dilakukan pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Surabaya, 9 November 2011

Materai

Rp. 6000,-

Daniel Kristianto Haryono

ABSTRAKSI

Keteraturan suatu kegiatan diukur dengan waktu. Setiap aktivitas manusia selalu ditentukan awal dan akhirnya dalam satuan waktu. Jam adalah suatu instrumen yang digunakan sebagai penunjuk waktu. Bagi penyandang tunanetra, penggunaan jam yang sering dijumpai di pasaran adalah sulit karena tidak memiliki *preview* suara. Tugas Akhir kali ini bertujuan untuk membuat jam digital berbasis *microcontroller* dengan *preview* suara untuk penyandang tunanetra. Tugas Akhir ini merupakan pengembangan dari teknologi yang telah dikembangkan oleh saudara Mustamu pada tahun 2008 dengan judul "Perancangan Jam Digital dengan *Seven Segment* dan *Preview* Suara". Sebagai fungsi tambahan akan disertakan perhitungan dan penjadwalan waktu sholat. Algoritma penentuan waktu sholat diambil dari penelitian saudari Sari yang berjudul "Portable Penunjuk Sholat Lima Waktu Menggunakan *Microcontroller* MCS'51 dan GPS" pada tahun 2008.

Jam digital ini dilengkapi dengan RTC (*Real Time Clock*) DS1307 sebagai pewaktu dan ISD25120 sebagai IC (*Integrated Circuit*) suara. Pembacaan data serta penulisan data pada RTC DS1307 dilakukan secara serial dengan protokol komunikasi I^2C (*Inter-Integrated Circuit*). Sedangkan keluaran suara oleh *earphone* dari ISD25120 akan diakses dengan metode *address*. Proses pengaturan parameter-parameter yang diperlukan oleh jam digital untuk beroperasi dilakukan dengan bantuan komputer melalui komunikasi serial.

Kata kunci: Jam, digital, *microcontroller*, RTC, ISD25120

KATA PENGANTAR

Puji syukur sebesar-besarnya kepada Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan proyek Tugas Akhir ini. Tugas Akhir berjudul "Rancang Bangun Jam untuk Tunanetra dengan Penunjuk Waktu Sholat Berbasis *Microcontroller*" ini disusun untuk menyelesaikan Program Studi S1 Sistem Komputer di Sekolah Tinggi Manajemen Informatika dan Teknik Komputer Surabaya.

Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Papa, mama, kakakku, dan keponakanku yang selalu memperhatikan, mendukung, dan menjagaku dari dahulu, sekarang, dan sampai selamanya
2. Bapak Harianto, S.Kom, M.Eng dan Bapak Madha Christian Wibowo, S.Kom selaku Dosen Pembimbing I dan Dosen Pembimbing II yang telah sabar membimbing, memberikan motivasi dan saran, terutama selama proses penyusunan Tugas Akhir ini
3. Dosen Waliku yang luar biasa, Bapak Yuwono Marta Dinata, M.Eng
4. Bapak Jimmy selaku sumber wawancara yang telah rela menyediakan waktu
5. Para dosen dan karyawan STIKOM Surabaya yang banyak sekali memberikan bantuan, baik yang saya sadari maupun tidak
6. Saudara-saudara seangkatan dan seperjuanganku, mahasiswa S1-SK angkatan 2007 atas kebersamaan yang indah
7. Seluruh pendidikku yang kuhormati
8. Sahabat-sahabatku yang kukasihi

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Semua manusia tidak pernah lepas dari waktu. Kita selalu menentukan awal atau akhir aktivitas kita dengan satuan waktu tertentu. Penentuan waktu dimaksudkan agar kita mempunyai kesepakatan bersama sehingga segala aktivitas kita berjalan secara teratur dan berkesinambungan. Hampir semua jam yang selama ini dijumpai di pasaran tidak memiliki *preview* suara untuk antarmukanya dengan pengguna. Hal ini tentu sangat menyulitkan para tunanetra dalam mengetahui waktu aktual sekarang dikarenakan keterbatasan pengelihatan mereka.

Menurut wawancara dengan seorang penyandang tunanetra, Bapak Jimmy, yang juga adalah seorang tenaga pengajar YPAB (Yayasan Pendidikan Anak-anak Buta), para tunanetra mempunyai ketergantungan yang cukup tinggi terhadap suara. Tidak semua informasi dapat diwujudkan dengan mudah kedalam bentuk *Braille*, seperti informasi pada layar komputer, informasi pada layar *handphone*, dan informasi waktu pada jam.

Teknologi saat ini yang dapat digunakan namun kurang tepat untuk menjawab permasalahan diatas telah dikembangkan oleh saudara Mustamu, dengan judul penelitian "Perancangan Jam Digital dengan *Seven Segment* dan *Preview* Suara" pada tahun 2008. Kekurangan alat tersebut apabila digunakan oleh para tunanetra terletak pada dimensi alat yang cukup besar dan penggunaan *seven segment* yang tidak tepat untuk tunanetra.

Berdasarkan dari permasalahan diatas maka dirancang sebuah jam dengan *preview* suara dengan dimensi yang lebih kecil. Sebagai fungsi tambahan, akan diberikan peringatan sholat lima waktu. Algoritma penentuan jadwal sholat lima waktu diambil dari penelitian saudari Sari yang berjudul "Portable Penunjuk Sholat Lima Waktu Menggunakan *Microcontroller* MCS'51 dan GPS" pada tahun 2008.

1.2. Perumusan Masalah

Berdasarkan dari latar belakang diatas maka dapat dirumuskan masalah yaitu bagaimana merancang jam digital berbasis *microcontroller* yang dapat melakukan perhitungan jadwal sholat lima waktu dan dapat memberikan informasi waktu aktual serta peringatan waktu sholat melalui media suara?

1.3. Pembatasan Masalah

Dalam perancangan dan pembuatan alat ini, terdapat beberapa pembatasan masalah, antara lain:

1. Bahasa yang digunakan pada *preview* suara adalah bahasa Indonesia
2. Pengaturan waktu dan parameter kota pilihan oleh pengguna akan menjadi sangat rumit dan tidak akurat, maka proses tersebut akan dilakukan dengan menggunakan bantuan komputer melalui komunikasi secara serial
3. Pengaturan parameter lintang suatu kota dengan nilai yang tinggi (> 40°) dapat menyebabkan kesalahan dalam perhitungan penjadwalan waktu sholat.

1.4. Tujuan

Tujuan dari pembuatan alat atau sistem ini adalah untuk merancang jam digital berbasis *microcontroller* yang dapat melakukan perhitungan jadwal sholat

lima waktu dan dapat memberikan informasi waktu aktual serta peringatan waktu sholat melalui media suara.

1.5. Kontribusi

Pada penelitian yang sebelumnya tentang perancangan jam digital dengan *seven segment* dan *preview suara* yang dibuat oleh Mustamu pada tahun 2008, sistem ini memiliki beberapa kekurangan apabila digunakan oleh para tunanetra yaitu dimensi alat yang besar dan penggunaan *seven segment* yang tidak tepat untuk tunanetra.

Dengan demikian, dibuatlah alat dengan dimensi yang lebih kecil dengan tujuan agar dapat dibawa kemana-mana oleh para tunanetra serta disertakan peringatan sholat lima waktu sebagai fungsi tambahannya. Penelitian ini berjudul "Rancang Bangun Jam untuk Tunanetra dengan Penunjuk Waktu Sholat Berbasis *Microcontroller*" dengan harapan supaya berguna untuk meningkatkan pelayanan publik terutama untuk para tunanetra. Tetapi seperti halnya penelitian lainnya, penelitian ini masih jauh dari sempurna. Untuk itu harus lebih disempurnakan lagi dari kekurangan-kekurangan yang ada.

1.6. Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun atas beberapa bab dengan urutan sebagai berikut :

BAB I : Pendahuluan

Pada bab ini diuraikan mengenai latar belakang dari topik tugas akhir yang diambil, kemudian dirumuskan menjadi suatu permasalahan yang akan diselesaikan dalam tugas akhir ini, batasan-batasan

masalah yang akan diteliti, tujuan dari penelitian tugas akhir ini, kontribusi yang dapat diberikan dari hasil penelitian ini terhadap perkembangan ilmu pengetahuan khususnya penerapan ilmu komputer dalam hal pelayanan pubilk, serta sistematika penulisan buku Tugas Akhir.

BAB II : Landasan Teori

Bagian landasan teori ini menguraikan tentang teori-teori yang terkait dengan variabel-variabel penelitian termasuk uraian tentang pemilihan suatu teori yang diterapkan dalam menyelesaikan masalah. Teori yang akan diuraikan adalah tentang *microcontroller* ATmega32, ISD25120, RTC DS1307, dan penentuan waktu sholat lima waktu.

BAB III : Metode Penelitian

Dalam bab ini diuraikan tentang metode penelitian yang digunakan dalam penelitian ini serta alasan penggunaan metode tersebut dalam penelitian. Pada metode penelitian ini dimuat perancangan perangkat lunak serta pembuatan yaitu proses analisa sesuai dengan *listing* program.

BAB IV : Pengujian dan Evaluasi Sistem

Dalam bagian pengujian dan evaluasi sistem ini diuraikan tentang prosedur dan hasil-hasil pengujian serta analisa hasil percobaan atau penelitian. Pada bagian ini dimuat tentang prosedur penelitian, hasil pengujian serta analisa hasil pengujian.

BAB V : Penutup

Bagian ini merupakan bagian akhir dari laporan penelitian tugas akhir ini yang menguraikan kesimpulan-kesimpulan yang diperoleh dari proses penelitian serta saran-saran untuk pengembangan penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1. *Microcontroller* ATmega32

Microcontroller dan *microprocessor* mempunyai beberapa perbedaan. *Microprocessor* yang terdapat pada komputer seperti Intel Pentium, hanya dapat bekerja apabila terdapat komponen pendukung seperti RAM (*Random Access Memory*), *hard disk*, *motherboard*, perangkat I/O, dll. Komponen-komponen tersebut diperlukan karena *microprocessor* hanya dapat melakukan pengolahan data, namun tidak dapat menyimpan data, menyimpan program, menerima masukan dari *user* secara langsung, ataupun menyampaikan data hasil pemrosesan ke keluaran. Berbeda dengan *microprocessor*, *microcontroller* sudah dilengkapi dengan komponen-komponen yang dikemas dalam satu *chip* seperti memori, perangkat I/O, *timer*, ADC (*Analog to Digital Converter*), dll. Hal ini membuat *microcontroller* lebih tepat untuk digunakan pada aplikasi *embedded system*. (Husanto, 2008)

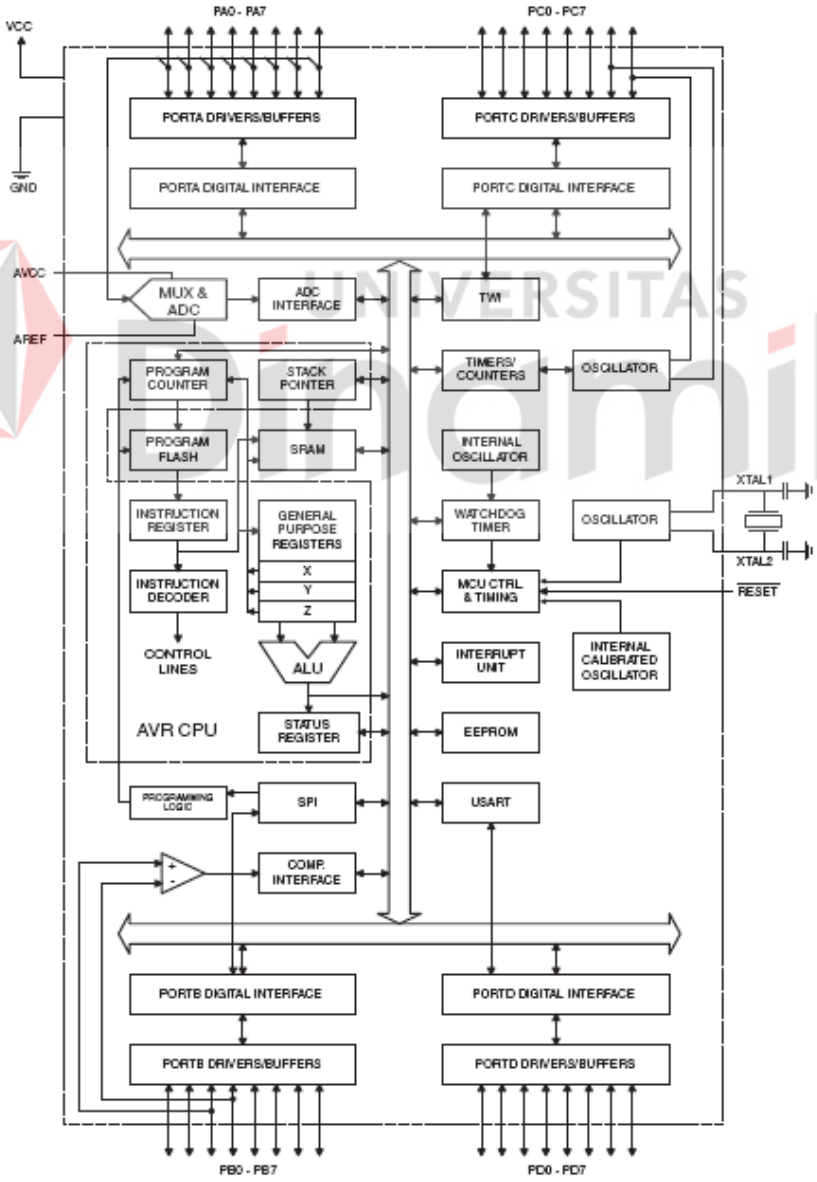
Microcontroller yang digunakan pada proyek ini adalah *microcontroller* keluarga AVR yang mempunyai arsitektur 8-bit RISC (*Reduce Instruction Set Compute*) produksi ATMEL yaitu ATmega32. Salah satu kelebihan arsitektur RISC dari arsitektur CISC (*Complex Instruction Set Compute*) adalah kecepatan waktu eksekusi tiap instruksi. Sebagian besar instruksi RISC dieksekusi dalam waktu satu *clock cycle*, sedangkan pada CISC sebagian besar instruksi dieksekusi dalam waktu dua belas *clock cycle*.

Beberapa fitur yang dimiliki ATmega32 adalah sebagai berikut (ATMEL, 2011):

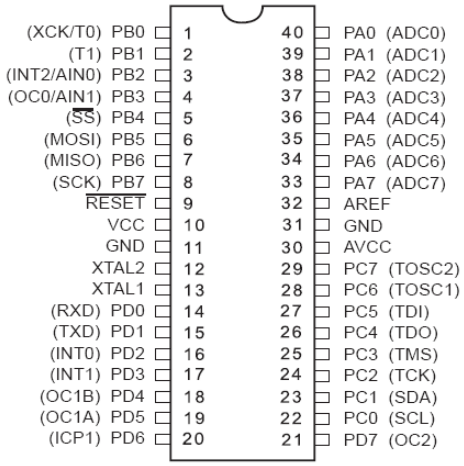
a. Mempunyai kinerja tinggi dengan konsumsi daya yang rendah
b. *Fully static operation*
c. Kinerja mencapai 16 MIPS (*Millions Instruction per Seconds*) pada osilator dengan nilai frekuensi 16 MHz
d. Memiliki kapasitas memori *Flash* sebesar 32 kByte, EEPROM (*Electrically Erasable Programmable Read-Only Memory*) sebesar 1024 Byte, dan SRAM (*Static Random-Access Memory*) sebesar 2 kByte
e. Memiliki 32 jalur I/O
f. Memiliki 2 buah *Timer*/*Counter* 8-bit dan 1 buah *Timer*/*Counter* 16-bit
g. Memiliki 4 kanal PWM (*Pulse Width Modulation*)
h. Memiliki 8 kanal ADC 10-bit
i. Memiliki antarmuka: *Two-wire Serial Interface*, USART (*Universal Synchronous Asynchronous Receiver*/*Transmitter*), SPI (*Serial Peripheral Interface Bus*)
j. Memiliki *Watchdog Timer* dengan osilator internal yang terpisah
k. Memiliki *Comparator* tegangan analog
l. Memiliki unit interupsi eksternal dan internal
m. Bekerja pada tegangan 4.5 V – 5.5 V dengan konsumsi arus maksimal 15 mA (dengan osilator 8 MHz, tegangan 5 V dan suhu pada rentang -40 °C - 85 °C).

Proses pemrograman ATmega32 dilakukan menggunakan fitur ISP (*In-System Programmable*) melalui antarmuka SPI (*Serial Peripheral Interface*). Fitur ISP memungkinkan untuk melakukan proses *download* program ke dalam

microcontroller tanpa bantuan *microcontroller master* seperti proses *download* program pada *microcontroller* AT89C51. *File* dengan ekstensi ".hex", yaitu kode program yang telah di-*compile* akan dikirimkan secara serial ke *microcontroller* untuk ditulis ke dalam memori *Flash* melalui jalur SPI yaitu *pin* MISO (*Master In Slave Out*), MOSI (*Master Out Slave In*), dan SCK (*Serial Clock*) yang digunakan sebagai sinyal sinkronasi komunikasi. Diagram blok ATmega32 terdapat pada Gambar 2.1, sedangkan konfigurasi pin ATmega32 terdapat pada Gambar 2.2.



Gambar 2.1 Blok diagram ATmega32 (ATMEL, 2011)



Gambar 2.2 Konfigurasi pin ATmega32 (ATMEL, 2011)

2.1.1. Fungsi–fungsi Pin pada ATmega32

a. VCC : Sumber tegangan +5V DC (*Direct Current*). (pin 10)

b. GND : Pin yang dihubungkan dengan *ground* sebagai referensi untuk VCC. (pin 11 dan pin 31)

c. *Port* A (PA0..PA7) merupakan pin I/O dua arah dan pin masukan tegangan analog untuk ADC

d. *Port* B (PB0..PB7) merupakan pin I/O dua arah dengan fungsi alternatif, seperti yang terlihat pada Tabel 2.1 di bawah

Tabel 2.1 Fungsi alternatif *Port* B

Pin	*Alternate Functions*
PB7	SCK (SPI *Bus Serial Clock*)
PB6	MISO (SPI *Bus Master Input*/*Slave Output*)
PB5	MOSI (SPI *Bus Master Output*/*Slave Input*)
PB4	$\overline{SS}$ (SPI *Slave Select Input*)
PB3	AIN1 (*Analog Comparator Negative Input*) OC0 (*Timer*/*Counter*0 *Output Compare Match Output*)
PB2	AIN0 (*Analog Comparator Positive Input*) INT2 (*External Interrupt* 2 *Input*)
PB1	T1 (*Timer*/*Counter*1 *External Counter Input*)
PB0	T0 (*Timer*/*Counter*0 *External counter Input*) XCK (USART *External Clock Input*/*Output*)

Sumber: ATMEL (2011)

e. *Port* C (PC0..PC7) merupakan pin I/O dua arah dengan fungsi alternatif, seperti yang terlihat pada Tabel 2.2 di bawah

Tabel 2.2 Fungsi alternatif *Port* C

Pin	*Alternate Functions*
PC7	TOSC2 (*Timer Oscillator Pin* 2)
PC6	TOSC1 (*Timer Oscillator Pin* 1)
PC5	TDI (JTAG *Test Data In*)
PC4	TDO (JTAG *Test Data Out*)
PC3	TMS (JTAG *Test Mode Select*)
PC2	TCK (JTAG *Test Clock*)
PC1	SDA (*Two-wire Serial Bus Data Input*/*Output Line*)
PC0	SCL (*Two-wire Serial Bus Clock Line*)

Sumber: ATMEL (2011)

f. *Port* D (PD0..PD7) merupakan pin I/O dua arah dengan fungsi alternatif, seperti yang terlihat pada Tabel 2.3 di bawah

Tabel 2.3 Fungsi alternatif *Port* D

Pin	*Alternate Functions*
PD7	OC2 (*Timer*/*Counter*2 *Output Compare Match output*)
PD6	ICP (*Timer*/*Counter*1 *Input Capture*)
PD5	OC1A (*Timer*/*Counter*1 *Output Compare* A *Match Output*)
PD4	OC1B (*Timer*/*Counter*1 *Output Compare* B *Match Output*)
PD3	INT1 (*External Interrupt* 1 *Input*)
PD2	INT0 (*External Interrupt* 0 *Input*)
PD1	TXD (USART *Output Pin*)
PD0	RXD (USART *Input Pin*)

Sumber: ATMEL (2011)

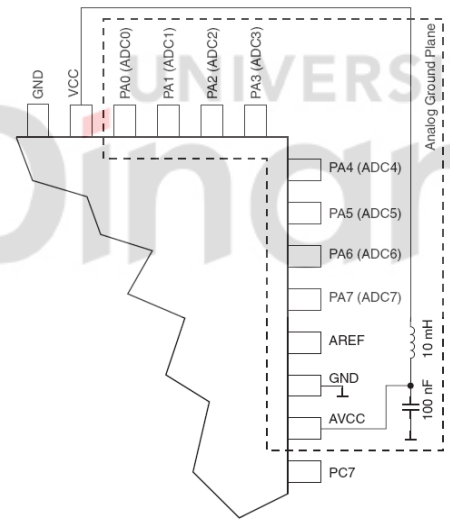
g. $\overline{RESET}$: Masukan untuk *reset* (*active low*). Jika diberikan kondisi *low* paling tidak selama 1.5 µS akan menghasilkan kondisi *reset* pada *microcontroller* meskipun *microcontroller* tidak mendapat *clock* dari osilator. (pin 9)

h. XTAL1 : Masukan ke penguat osilator. Pin ini dihubungkan dengan kristal atau sumber osilator yang lain. (pin 13)

i. XTAL2 : Keluaran dari penguat osilator. Pin ini dihubungkan dengan kristal atau *ground*. (pin 12)

j. AVCC : Pin yang digunakan untuk memberikan sumber tegangan pada *Port* A. Pin ini harus tetap dihubungkan dengan VCC meskipun fitur ADC tidak digunakan. Apabila fitur ADC digunakan, maka pin AVCC harus dihubungkan dengan VCC melalui *low-pass filter* seperti yang terlihat pada Gambar 2.3. (pin 30)

k. AREF : Pin yang digunakan sebagai masukan tegangan referensi untuk ADC. (pin 32).

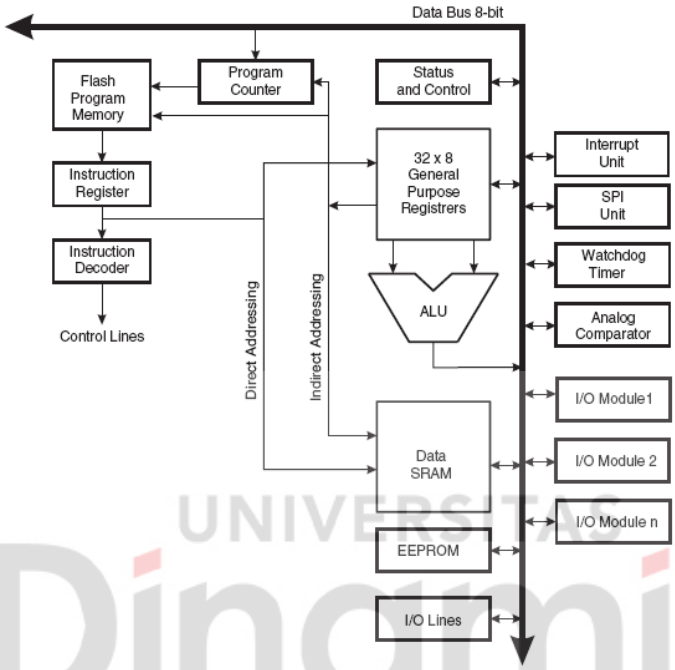


Gambar 2.3 Koneksi AVCC dengan VCC melalui *low-pass filter* (ATMEL, 2011)

2.1.2. Arsitektur CPU (*Central Processing Unit*) AVR

Agar kinerja yang dihasilkan maksimal, *microcontroller* AVR menggunakan arsitektur *Harvard* yang memisahkan antara memori dan *bus* untuk program dan data. Instruksi yang terdapat pada memori program dieksekusi dengan metode *single level pipelining*. Pada saat suatu instruksi akan dieksekusi,

instruksi berikutnya sudah siap untuk diambil dari memori program. Konsep ini akan meyakinkan bahwa pada setiap *clock cycle* akan terdapat instruksi untuk dieksekusi oleh *microcontroller*. Diagram blok dari arsitektur *microcontroller* AVR terdapat pada Gambar 2.4. (ATMEL, 2011)



Gambar 2.4 Diagram blok arsitektur *microcontroller* AVR (ATMEL, 2011)

2.1.2.1. *Arithmetic Logic Unit* (ALU)

ALU terhubung secara langsung dengan 32 *General Purpose Working Register*. Dalam satu *clock cycle*, ALU dapat melakukan operasi aritmatik antara dua buah register atau antara register dengan *immediate*. Operasi dari ALU dapat dibedakan menjadi tiga kategori – aritmatik, logika, dan *bit-function*. (ATMEL, 2011)

2.1.2.2. *Status Register* (SREG)

Status Register berisi informasi mengenai hasil dari instruksi aritmatik yang baru saja selesai dieksekusi. Informasi pada SREG dapat digunakan untuk mengubah alur program, yaitu dengan cara membuat operasi berkondisi tertentu. SREG tidak akan disimpan saat akan menjalankan rutin interupsi ataupun dikembalikan pada nilai awal saat rutin interupsi selesai dieksekusi. Susunan dari SREG terdapat pada Tabel 2.4.

Tabel 2.4 *Status Register* AVR

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

a. Bit 7 : I (*Global Interrupt Enable*)

Bit untuk mengaktifkan interupsi. Bit I akan di-*clear* oleh *hardware* jika terjadi *interrupt* dan akan di-*set* kembali oleh instruksi RETI. Bit I dapat di-*set* secara manual melalui program menggunakan instruksi SEI atau di-*clear* menggunakan instruksi CLI

b. Bit 6 : T (*Bit Copy Storage*)

c. Bit 5 : H (*Half Carry Flag*)

d. Bit 4 : S (*Sign Bit*)

Bit S merupakan hasil operasi EOR (*Exclusive OR*) antara N (*Negative Flag*) dengan V (*Two's Complement Overflow Flag*)

e. Bit 3 : V (*Two's Complement Overflow Flag*)

Bit yang berguna untuk mendukung operasi aritmatika

f. Bit 2 : N (*Negative Flag*)

Bit N akan di-*set* apabila hasil operasi suatu instruksi adalah bilangan negatif

g. Bit 1 : Z (*Zero Flag*)

Bit Z akan di-*set* apabila hasil operasi suatu instruksi adalah nol

h. Bit 0 : C (*Carry Flag*)

Bit C akan di-*set* apabila hasil operasi suatu instruksi menghasilkan *carry*.

2.1.2.3. *General Purpose Register* (GPR)

GPR adalah register kerja (R0 - R31) yang mempunyai ruangan 8-bit. GPR berfungsi sebagai tempat ALU mengeksekusi kode-kode program, setiap instruksi ALU melibatkan GPR. Susunan dari GPR AVR terdapat pada Tabel 2.5. GPR dibagi menjadi dua yaitu kelompok atas (R16 - R31) dan kelompok bawah (R0 - R15), dimana kelompok bawah tidak dapat digunakan untuk operasi instruksi yang melibatkan *immediate* seperti instruksi LDI (*Load Immediate*), dan hanya dapat digunakan untuk operasi yang melibatkan antar register, SRAM, atau register I/O (register *port*). Kelompok atas mempunyai fungsionalitas yang sama dengan kelompok bawah hanya saja mempunyai kelebihan dapat digunakan untuk operasi yang melibatkan *immediate*. R27 - R31 merupakan register pasangan yang digunakan untuk *pointer* (penunjuk ke alamat tertentu). XH:XL (R27:R26), YH:YL (R29:R28), ZH:ZL (R31:R30), hanya register *pointer* Z yang dapat digunakan untuk menunjuk ke alamat memori program. (Winoto, 2008)

Tabel 2.5 *General Purpose Register* AVR

	7 0	Addr.	
General Purpose Working Register	R0	$00	
	R1	$01	
	R2	$02	
	…		
	R13	$0D	
	R14	$0E	
	R15	$0F	
	R16	$10	
	R17	$11	
	…		
	R26	$1A	X-*register Low Byte*
	R27	$1B	X-*register High Byte*
	R28	$1C	Y-*register Low Byte*
	R29	$1D	Y-*register High Byte*
	R30	$1E	Z-*register Low Byte*
	R31	$1F	Z-*register High Byte*

Sumber: ATMEL (2011)

2.1.2.4. *Stack Pointer*

Stack digunakan untuk menyimpan data sementara, variable lokal, dan *return address* setelah melakukan pemanggilan subrutin atau interupsi. *Stack pointer* selalu menunjuk pada puncak *stack*. *Stack* diimplementasikan mulai dari lokasi memori tertinggi menuju ke lokasi memori terendah, sehingga penggunaan perintah PUSH akan mengurangi nilai *stack pointer*.

Sebelum menggunakan *stack* perlu dilakukan pendefinisian pada program, yaitu alamat SRAM yang akan digunakan sebagai *stack*. Pendefinisian harus dilakukan terlebih dahulu sebelum melakukan pemanggilan subrutin atau interupsi. Nilai yang diisikan pada *stack pointer* pada saat pendefinisian harus di atas $60, hal ini berkaitan dengan pemetaan memori pada *microcontroller* AVR. Nilai *stack pointer* akan berkurang sebanyak satu pada saat suatu data dimasukkan (dengan perintah PUSH) pada *stack*, dan akan berkurang sebanyak dua pada saat

return address dimasukkan pada stack, yaitu pada saat pemanggilan subrutin atau interupsi. Susunan dari *stack pointer* AVR terdapat pada Tabel 2.6.

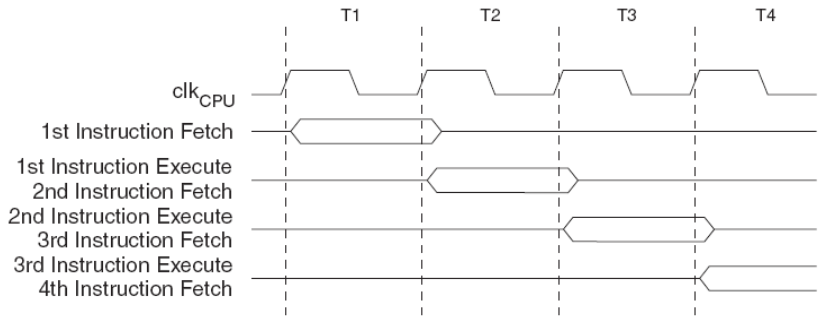
Tabel 2.6 *Stack Pointer* AVR

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

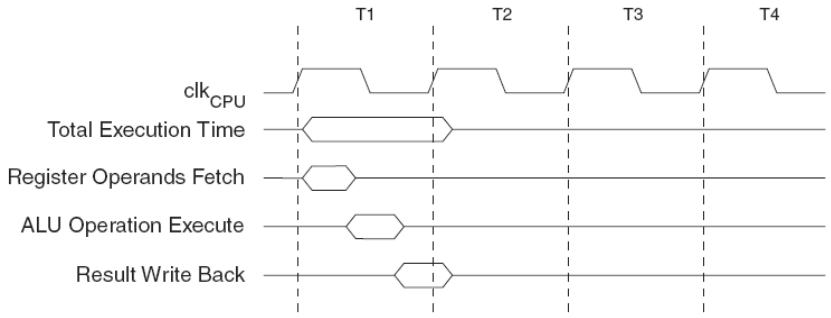
Sumber: ATMEL (2011)

2.1.2.5. *Instruction Execution Timing*

AVR menggunakan arsitektur *Harvard* yang memisahkan antara memori dan *bus* untuk program dan data, sehingga pengambilan instruksi dan eksekusi instruksi dapat dilakukan secara paralel. Gambar 2.5 menunjukkan kinerja AVR yang mengimplementasikan arsitektur *Harvard* dan konsep *fast-access Register File*. Sedangkan pada Gambar 2.6 ditunjukkan operasi ALU dalam satu *clock cycle*.



Gambar 2.5 Pengambilan instruksi dan eksekusi instruksi AVR (ATMEL, 2011)



Gambar 2.6 Operasi ALU dalam satu *clock cycle* (ATMEL, 2011)

2.1.2.6. *Reset* dan *Interrupt Handling*

Alamat terendah pada program memori secara default didefinisikan sebagai *reset* dan vektor interupsi. Daftar dari vektor interupsi terdapat pada Tabel 2.7. Semakin kecil nilai alamat dari vektor interupsi maka prioritas interupsi tersebut untuk dijalankan terlebih dahulu semakin tinggi.

Saat suatu interupsi terjadi, *Global Interrupt Enable* (bit ke-7 pada SREG) akan di-*clear* sehingga semua interupsi yang terjadi sesudah interupsi pertama tidak akan dieksekusi sebelum interupsi pertama selesai. *Global Interrupt Enable* dapat di-*set* secara manual melalui program untuk mengijinkan interupsi bersarang, sehingga semua interupsi yang diaktifkan dan terjadi sebelum interupsi pertama selesai dapat dieksekusi terlebih dahulu. *Global Interrupt Enable* secara otomatis akan di-*set* pada saat instruksi RETI telah dieksekusi. Saat AVR keluar dari rutin interupsi, AVR akan kembali ke program utama lalu mengeksekusi satu instruksi sebelum kembali melayani antrian interupsi.

Tabel 2.7 Vektor *reset* dan *interrupt*

Vector No.	*Program Address*	*Source*	*Interrupt Definition*
1	$000	RESET	*External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset,* dan JTAG (*Join Test Action Group*) AVR *Reset*
2	$002	INT0	*External Interrupt Request* 0
3	$004	INT1	*External Interrupt Request* 1
4	$006	INT2	*External Interrupt Request* 2
5	$008	TIMER2 COMP	*Timer/Counter*2 *Compare Match*
6	$00A	TIMER2 OVF	*Timer/Counter*2 *Overflow*
7	$00C	TIMER1 CAPT	*Timer/Counter*1 *Capture Event*
8	$00E	TIMER1 COMPA	*Timer/Counter*1 *Compare Match* A
9	$010	TIMER1 COMPB	*Timer/Counter*1 *Compare Match* B
10	$012	TIMER1 OVF	*Timer/Counter*1 *Overflow*
11	$014	TIMER0 COMP	*Timer/Counter*0 *Compare Match*
12	$016	TIMER0 OVF	*Timer/Counter*0 *Overflow*
13	$018	SPI, STC	*Serial Transfer Complete*
14	$01A	USART, RXC	USART, Rx *Complete*
15	$01C	USART, UDRE	USART *Data Register Empty*
16	$01E	USART, TXC	USART, Tx *Complete*
17	$020	ADC	ADC *Conversion Complete*
18	$022	EE_RDY	EEPROM *Ready*
19	$024	ANA_COMP	*Analog Comparator*
20	$026	TWI	*Two-wire Serial Interface*
21	$028	SPM_RDY	*Store Program Memory Ready*

Sumber: ATMEL (2011)

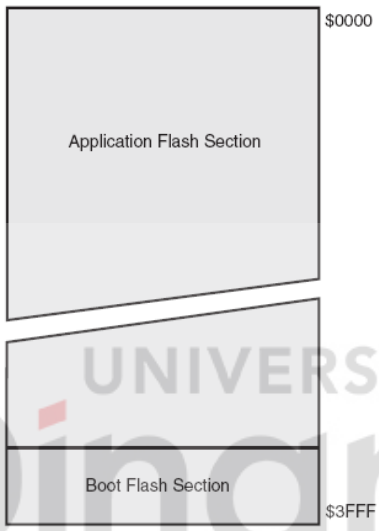
2.1.3. Memori ATmega32

Arsitektur AVR mempunyai dua memori utama, yaitu memori data (SRAM) dan memori program (*Flash*). Selain SRAM dan *Flash*, ATmega32 memiliki fitur tambahan, yaitu memori EEPROM yang dapat digunakan untuk penyimpanan data yang bersifat *non-volatile*.

2.1.3.1. Memori *Flash*

Adalah memori ROM (*Read-Only Memory*) tempat kode-kode program berada. Kata *Flash* menunjukkan jenis ROM yang dapat ditulis dan dihapus secara elektrik. Memori *Flash* terbagi menjadi dua bagian yaitu bagian aplikasi

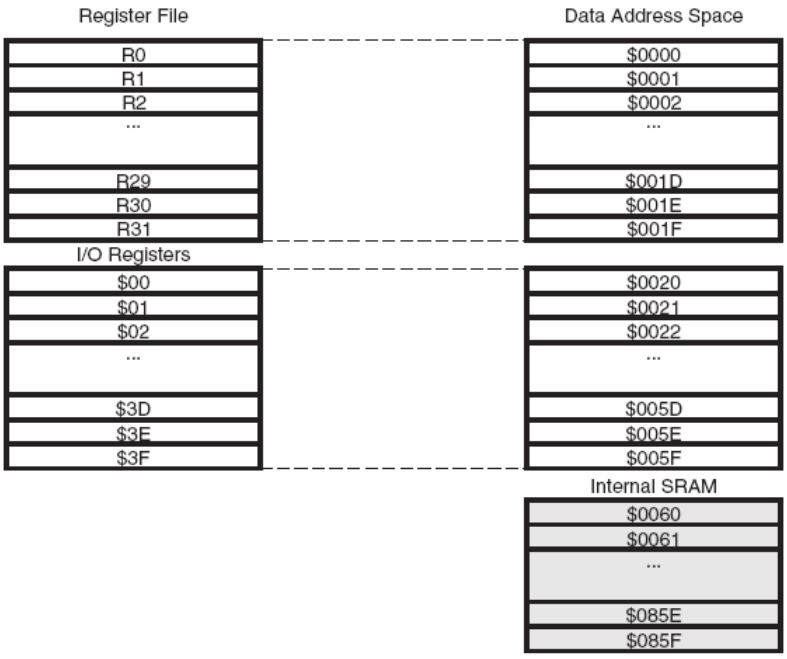
dan bagian *boot*. Bagian aplikasi adalah bagian kode-kode program aplikasi berada. Sedangkan bagian *boot* adalah bagian yang digunakan untuk *booting* awal yang dapat diprogram untuk menulis bagian aplikasi tanpa melalui *programmer*/*downloader*, misalnya melalui USART. Apabila bagian *boot* tidak digunakan, maka bagian tersebut dapat digunakan untuk bagian aplikasi. Gambar 2.7 menunjukkan peta memori *Flash* pada ATmega32. (Winoto, 2008)



Gambar 2.7 Peta memori *Flash* ATmega32 (ATMEL, 2011)

2.1.3.2. Memori Data

Memori data ATmega32 dibagi menjadi tiga bagian, yaitu 32 Byte *General Purpose Register*, 64 Byte register I/O, dan 2048 Byte SRAM internal. *General Purpose Register* menempati alamat $0000 sampai $001F, register I/O menempati alamat $0020 sampai $005F, sedangkan SRAM internal menempati alamat $0060 sampai $085F. Register I/O merupakan register yang digunakan untuk mengatur fungsi-fungsi berbagai *peripheral microcontroller* seperti *Control Register, Timer*/*Counter*, fungsi-fungsi I/O, dsb. Gambar 2.8 menunjukkan peta memori data pada ATmega32.



Gambar 2.8 Peta memori data pada ATmega32 (ATMEL, 2011)

2.1.3.3. EEPROM

ATmega32 memiliki memori data EEPROM sebesar 1024 Byte. EEPROM disusun dengan ruang data terpisah dimana dapat dilakukan proses baca atau tulis untuk data sebesar 1 Byte. EEPROM pada ATmega32 memiliki ketahanan untuk proses *read*/*write* sampai 100.000 kali.

Penggunaan EEPROM melibatkan tiga buah *register* yaitu EEPROM *Address Register* (EEAR), EEPROM *Data Register* (EEDR), dan EEPROM *Control Register* (EECR). Penjelasan dari register-register tersebut adalah sebagai berikut:

A. EEPROM *Address Register*

Tabel 2.8 EEPROM *Address Register* ATmega32

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 15..10 - *Reserved Bits*: bit yang digunakan untuk keperluan internal ATmega32. Selalu bernilai nol

b. Bit 9..0 - EEPROM *Address*: bit yang merepresentasikan alamat dari EEPROM. EEPROM dialamatkan secara linear mulai dari nilai 0 sampai dengan 1023. Sebelum melakukan operasi pada EEPROM, nilai dari EEAR harus didefinisikan terlebih dahulu.

B. EEPROM *Data Register*

Tabel 2.9 EEPROM *Data Register* ATmega32

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Keterangan:

Bit 7..0 - EEPROM *Data*: EEDR akan berisi data yang akan ditulis atau dibaca pada alamat yang sesuai dengan nilai EEAR.

C. EEPROM *Control Register*

Tabel 2.10 EEPROM *Control Register* ATmega32

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 7..4 - *Reserved Bits*: bit yang digunakan untuk keperluan internal ATmega32. Selalu bernilai nol

b. Bit 3 - EEPROM *Ready Interrupt Enable*: apabila EERIE di-*set*, maka interupsi EERPOM *Ready* akan diaktifkan jika bit I pada SREG di-*set*. Saat bit EEWE bernilai nol, *interrupt* EEPROM *Ready* akan memberikan sinyal *interrupt* secara terus menerus

c. Bit 2 - EEPROM *Master Write Enable*: bit EEMWE akan menentukan apakah memberikan nilai 1 pada EEWE menulis data pada EERPOM atau tidak. Pada saat EEMWE di-*set*, memberikan nilai 1 pada bit EEWE dalam waktu empat *clock cycle* akan menulis data pada EEPROM sesuai alamat yang telah ditentukan. Saat EEMWE di-*set* oleh program, EEMWE akan di-*clear* oleh *hardware* secara otomatis dalam waktu empat *clock cycle*

d. Bit 1 - EEPROM *Write Enable*: saat alamat dan data yang diperlukan untuk penulisan EEPROM sudah diberikan oleh program, memberikan nilai satu pada bit EEWE akan memulai penulisan EEPROM. Bit EEMWE harus di-*set* terlebih dahulu sebelum memberikan nilai satu pada bit EEWE

e. Bit 0 - EEPROM *Read Enable*: saat alamat yang diperlukan untuk melakukan operasi baca EEPROM sudah diberikan oleh program, memberikan nilai satu

pada bit EERE akan memulai operasi baca. Apabila operasi penulisan EEPROM masih berlangsung, pembacaan data dari EEPROM atau mengganti nilai *register* EEAR tidak dimungkinkan.

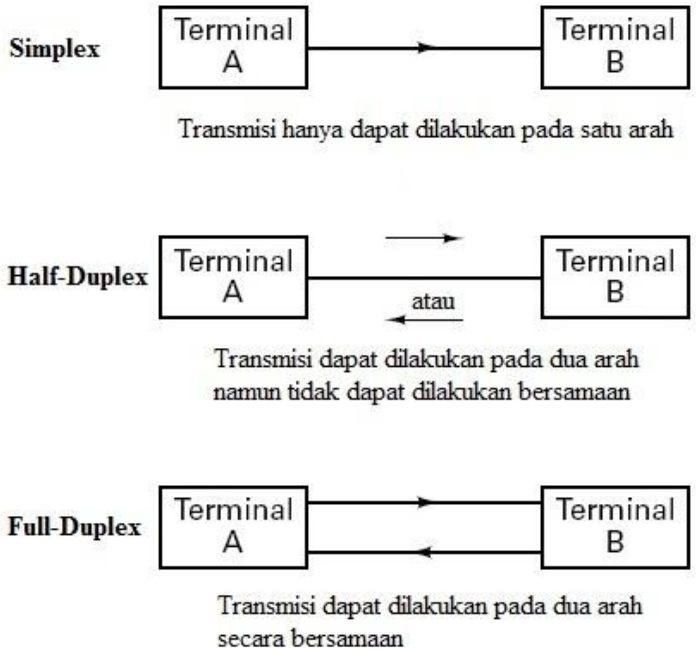
2.1.4. USART

Menurut Winoto (2008) USART dapat difungsikan sebagai transmisi data sinkron dan asinkron. Sinkron berarti *transmitter* dan *receiver* mempunyai satu sumber *clock* yang sama. Sedangkan asinkron berarti *transmitter* dan *receiver* yang mempunyai sumber *clock* yang berbeda.

Menurut Mazidi (2000) transmisi data secara serial adalah transmisi data dimana data tersebut akan dikirimkan sebanyak satu bit dalam satu satuan waktu. Terdapat dua cara dalam mentransmisikan data secara serial, yaitu secara *synchronous* dan *asynchronous*. Perbedaan dari kedua cara tersebut adalah sinyal *clock* yang dipakai sebagai sinkronisasi pengiriman data.

Transmisi secara *synchronous* yaitu pengiriman data serial yang disertai dengan sinyal *clock*, sedangkan *asynchronous* yaitu pengiriman data serial yang tidak disertai sinyal *clock* sehingga *receiver* harus membangkitkan sinyal *clock* sendiri (tidak memerlukan sinkronisasi). (Nalwan, 2003)

Pengiriman data secara serial dapat dibagi menjadi tiga menurut arah datanya, yaitu *Simplex*, *Half-Duplex* dan *Full-Duplex*. Ketiga mode tersebut diilustrasikan pada Gambar 2.9. (Mazidi, 2000)

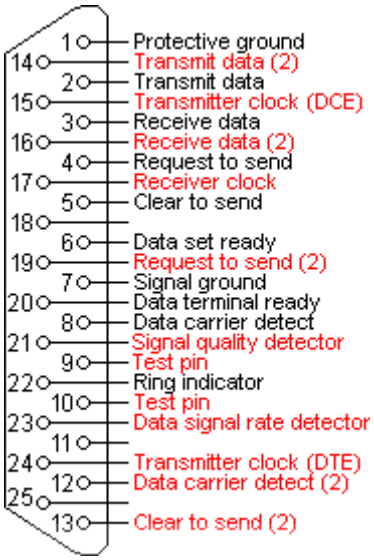


Gambar 2.9 Arah komunikasi serial (Lohala, 2011)

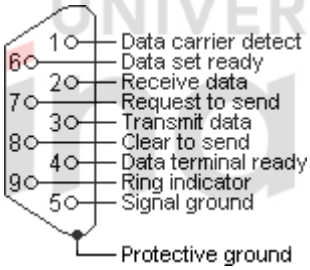
Satuan kecepatan transfer data (*baud rate*) pada komunikasi serial adalah bps (*bits per second*). Untuk menjaga kompatibilitas dari beberapa peralatan komunikasi data yang dibuat oleh beberapa pabrik, pada tahun 1960 EIA (*Electronics Industries Association*) melakukan standarisasi antarmuka serial dengan nama RS232.

Keluaran yang dihasilkan oleh RS232 tidak sesuai dengan keluaran TTL (*Transistor-Transistor Logic*) yang sudah ada. Dalam RS232, logika 1 direpresentasikan dengan tegangan -3 V sampai dengan -25 V sedangkan logika 0 direpresentasikan dengan tegangan +3 V sampai dengan +25 V. Hasil tak terdefinisi jika berada diantara tegangan -3 V sampai dengan +3 V. IBM PC atau komputer yang berbasis x86 (8086, 286, 386, 486, dan Pentium) secara umum *processor* yang digunakan memiliki dua *port* COM. Keduanya merupakan konektor jenis RS232 yaitu DB25 dan DB9. Ilustrasi DB25 dan keterangan

pinout-nya terdapat pada Gambar 2.10, sedangkan ilustrasi DB9 dan keterangan *pinout*-nya terdapat pada Gambar 2.11.

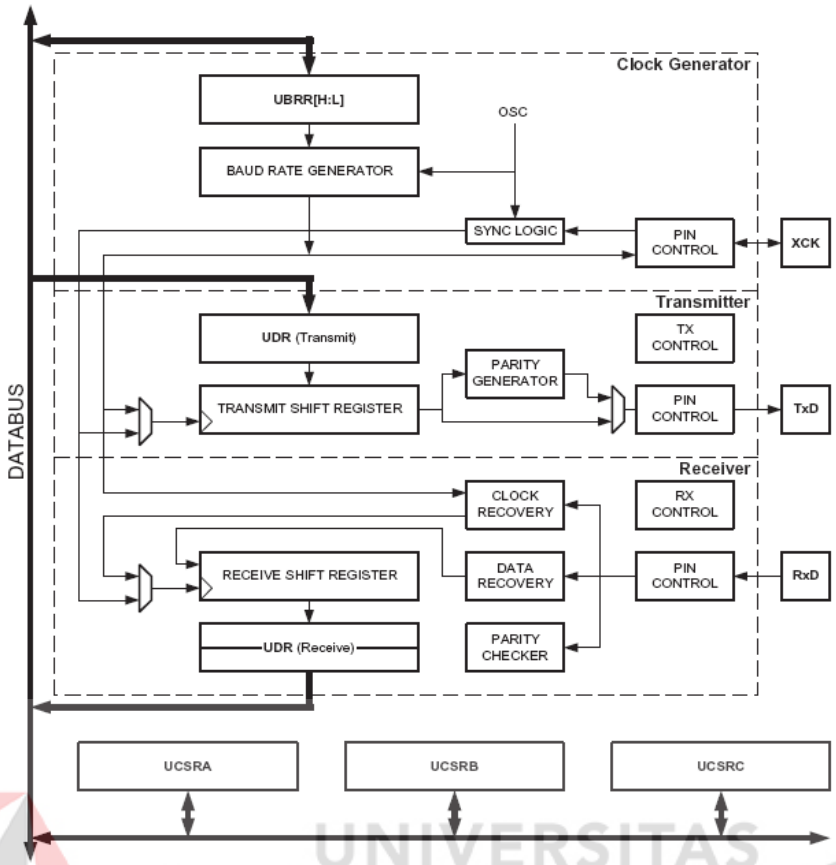


Gambar 2.10 *Pinout* konektor DB25 (Bies, 2011)



Gambar 2.11 *Pinout* konektor DB9 (Bies, 2011)

ATmega32 memiliki fitur antarmuka USART yang dapat digunakan untuk melakukan pengiriman data secara serial baik secara *synchronous* maupun *asynchronous*. Gambaran jelas diagram blok USART pada ATmega32 terdapat pada Gambar 2.13. Tiga bagian utama pada USART ATmega32 dipisahkan dengan kotak bergaris putus-putus yaitu, *Clock Generator*, *Transmitter*, dan *Receiver*.



Gambar 2.12 Diagram blok USART ATmega32 (ATMEL, 2011)

2.1.4.1. *Clock Generator*

Bagian yang berhubungan dengan kecepatan *baud rate*. Register yang berfungsi untuk menentukan besarnya *baud rate* adalah register UBRR. Tabel 2.11 menunjukkan persamaan yang digunakan untuk menentukan *baud rate* dan nilai dari register UBRR.

Tabel 2.11 Persamaan untuk menentukan nilai *baudrate* dan UBRR

Operating Mode	*Equation for Calculating Baud Rate*	*Equation for Calculating UBRR Value*
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{osc}}{16\,(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{16\,BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{osc}}{8\,(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{8\,BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{osc}}{2\,(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{2\,BAUD} - 1$

Sumber: ATMEL (2011)

Keterangan:

a. BAUD adalah kecepatan transfer bit yaitu *baud rate* dengan satuan bps
b. f_{osc} adalah nilai frekuensi dari osilator yang digunakan
c. UBRR merupakan gabungan dari dua register 8-bit yaitu UBRRH dan UBRRL, mempunyai rentang nilai antara 0 - 4095.

2.1.4.2. *Transmitter*

Transmitter dari USART diaktifkan dengan cara memberikan nilai satu pada bit TXEN (*Transmit Enable*) yang berada di register UCSRB. Saat *transmitter* diaktifkan maka fungsi normal pin TXD sebagai I/O akan dinonaktifkan dan fungsinya sebagai *transmitter* data serial akan diaktifkan. Pengaturan *baud rate*, mode operasi, dan *frame format* harus dilakukan sebelum mengaktifkan *transmitter*.

2.1.4.3. *Receiver*

Receiver dari USART diaktifkan dengan cara memberikan nilai satu pada bit RXEN (*Receive Enable*) yang berada di register UCSRB. Saat receiver diaktifkan maka fungsi normal pin RXD sebagai I/O akan dinonaktifkan dan fungsinya sebagai *receiver* data serial akan diaktifkan. Pengaturan *baud rate*, mode operasi, dan *frame format* harus dilakukan sebelum mengaktifkan *receiver*.

2.1.4.4. *Frame Formats*

Definisi *frame* dari suatu data serial adalah satu karakter data yang disertai dengan bit sinkronisasi (*start* dan *stop* bit), dan *parity* bit yang bersifat opsional. USART pada AVR mengijinkan 30 kombinasi *frame format* dari pilihan berikut:

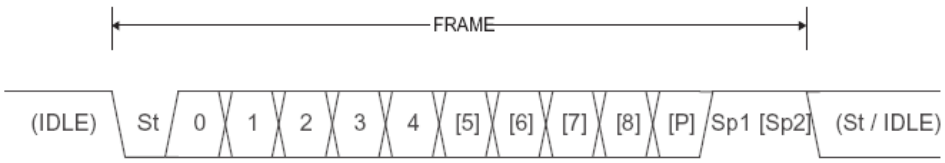
a. 1 start bit

b. 5, 6, 7, 8, atau 9 bit data

c. Dengan *even*, *odd*, atau tanpa *parity* bit

d. 1 atau 2 stop bit.

Sebuah *frame* selalu dimulai dengan start bit yang kemudian akan diikuti oleh bit data mulai dari bit terendah sampai bit tertinggi. Apabila menggunakan *parity* bit, maka bit tersebut akan diletakkan sesudah bit data dan sebelum stop bit. Gambar 2.13 mengilustrasikan susunan dari sebuah *frame*.



Gambar 2.13 *Frame format* (ATMEL, 2011)

Keterangan:

a. St: start bit, selalu berlogika *low*

b. (n): bit data (0 - 8)

c. P: *parity* bit

d. Sp: stop bit, selalu berlogika *high*

e. IDLE: tidak ada pertukaran data pada jalur komunikasi. Kondisi IDLE harus berlogika *high*.

2.1.4.5. Register Pengendali USART

Operasi dari USART dikendalikan oleh 5 register yaitu, USART *Data Register* (UDR), USART *Control and Status Register* A (UCSRA), USART *Control and Status Register* B (UCSRB), USART *Control and Status Register C*

(UCSRC), USART *Baud Rate Register* (UBRR). Penjelasan dari register-register tersebut adalah sebagai berikut:

A. USART *Data Register*

Tabel 2.12 USART *Data Register* ATmega32

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (*Read*)
	TXB[7:0]								UDR (*Write*)
Read/*Write*	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Register UDR digunakan sebagai *buffer* data yang akan dikirim dan *buffer* data yang diterima. UDR untuk *buffer transmitter* hanya dapat ditulis ketika bit/*flag* UDRE dalam register UCSRA bernilai satu. Apabila digunakan lebar data 5, 6, atau 7-bit, maka bit yang tidak digunakan tidak akan dihiraukan oleh *transmitter*, dan akan dijadikan nol saat diterima oleh *receiver*.

B. USART *Control and Status Register* A

Tabel 2.13 USART *Control and Status Register* A ATmega32

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/*Write*	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 7 - USART *Receive Complete*: bit ini akan di-*set* ketika data yang masuk pada UDR sudah lengkap dan belum dibaca. Bit ini akan di-*clear* ketika data pada UDR sudah dibaca atau UDR dalam keadaan kosong. Bit ini akan membangkitkan interupsi Rx *Complete* (jika diaktifkan) saat bernilai satu, dan

akan bernilai nol secara otomatis bersamaan dengan eksekusi interupsi yang bersangkutan

b. Bit 6 - USART *Transmit Complete*: bit ini akan di-*set* ketika data yang akan dikirim telah keluar. Bit ini akan membangkitkan interupsi Tx *Complete* (jika diaktifkan) saat bernilai satu, dan akan bernilai nol secara otomatis bersamaan dengan eksekusi interupsi yang bersangkutan
c. Bit 5 - USART *Data Register Empty*: bit ini digunakan sebagai indikator isi dari register UDR. Apabila bernilai satu maka UDR dalam keadaan kosong dan siap untuk diisi data berikutnya. Bit ini akan membangkitkan interupsi *Data Register Empty* (jika diaktifkan) saat bernilai satu, dan akan bernilai nol secara otomatis bersamaan dengan eksekusi interupsi yang bersangkutan
d. Bit 4 - *Frame Error*: bit ini digunakan sebagai indikator ketika data yang diterima *error*, misalnya ketika stop bit pertama data dibaca bernilai nol maka bit ini akan bernilai satu
e. Bit 3 - *Data OverRun*: bit ini digunakan sebagai indikator ketika terjadi data yang tumpang tindih (*overrun*). Bit ini akan bernilai satu apabila terjadi *overrun*
f. Bit 2 - *Parity Error*: bit ini digunakan untuk mendeteksi adanya *parity error*. Bit ini akan bernilai satu ketika didapati *parity* bit yang *error*
g. Bit 1 - *Double the* USART *Transmission Speed*: bit ini hanya dapat digunakan apabila menggunakan mode operasi *asynchronous*. Bit ini berfungsi untuk menggandakan *baudrate* yang digunakan

h. Bit 0 - *Multi-processor Communication Mode*: bit ini digunakan untuk mengaktifkan mode *Multi-processor Communication* yang akan mengabaikan data yang tidak memiliki informasi alamat.

C. USART *Control and Status Register* B

Tabel 2.14 USART *Control and Status Register* B ATmega32

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/*Write*	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 7 - RX *Complete Interrupt Enable*: nilai satu pada bit ini akan mengaktifkan interupsi Rx *Complete*. Ketika satu Byte data lengkap tanpa *error* diterima pada UDR, secara otomatis bit RXC pada register UCSRA akan di-*set*

b. Bit 6 - TX *Complete Interrupt Enable*: nilai satu pada bit ini akan mengaktifkan interupsi Tx *Complete*. Ketika satu Byte data lengkap tanpa *error* telah dikirim dari UDR, secara otomatis bit TXC pada register UCSRA akan di-*set*

c. Bit 5 - USART *Data Register Empty Interrupt Enable*: nilai satu pada bit ini akan mengaktifkan interupsi *Data Register Empty*. Ketika data yang ditulis pada UDR telah dikirimkan dan UDR dalam keadaan kosong maka sinyal interupsi *Data Register Empty* akan dibangkitkan

d. Bit 4 - *Receiver Enable*: bit ini digunakan untuk mengaktifkan fungsi pin RXD sebagai jalur penerimaan data USART

e. Bit 3 - *Transmitter Enable*: bit ini digunakan untuk mengaktifkan fungsi pin TXD sebagai jalur pengiriman data USART

f. Bit 2 - *Character Size*: bit ini digunakan bersama dengan UCSZ1 dan UCSZ0 pada register UCSRC yang berfungsi untuk memilih format lebar data bit (*Character Size*) yang digunakan

g. Bit 1 - *Receive Data Bit* 8: bit ini digunakan sebagai bit kedelapan ketika menggunakan format data 9-bit. Bit ini harus dibaca terlebih dahulu sebelum melakukan pembacaan data pada UDR

h. Bit 0 - *Transmit Data Bit* 8: bit ini digunakan sebagai bit kedelapan ketika menggunakan format data 9-bit. Penulisan pada bit ini harus dilakukan terlebuh dahulu sebelum melakukan penulisan pada UDR.

D. USART *Control and Status Register* C

Tabel 2.15 USART *Control and Status Register* C ATmega32

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 7 - *Register Select*: bit ini digunakan untuk memilih akses register UCSRC (dengan nilai satu) atau register UBRRH (dengan nilai nol) dikarenakan UCSRC dan UBRRH berbagi lokasi yang sama pada register I/O

b. Bit 6 - USART *Mode Select*: bit ini digunakan untuk memilih mode USART. Nilai satu untuk mode *synchronous* dan nilai nol untuk mode *asynchronous*

c. Bit 5:4 - *Parity Mode*: bit yang digunakan untuk memilih mode *parity* bit yang digunakan. Tabel 2.16 menunjukkan pengaturan mode *parity* bit menurut nilai UPM1 dan UPM0

Tabel 2.16 Pengaturan bit UPM untuk menentukan mode *parity* bit

UPM1	UPM0	*Parity Mode*
0	0	*Disabled*
0	1	*Reserved*
1	0	*Enabled, Even Parity*
1	1	*Enabled, Odd Parity*

Sumber: ATMEL (2011)

d. Bit 3 - *Stop Bit Select*: bit yang digunakan untuk memilih jumlah stop bit. Nilai satu untuk 1 stop bit dan nilai nol untuk 2 stop bit

e. Bit 2:1 - *Character Size*: bit yang digunakan untuk memilih lebar data bit yang digunakan. Digunakan bersama dengan bit UCSZ2. Tabel 2.17 menunjukkan pengaturan lebar data bit menurut nilai UCSZ2, UCSZ1, dan UCSZ0

Tabel 2.17 Pengaturan bit UCSZ untuk menentukan lebar data bit

UCSZ2	UCSZ1	UCSZ0	*Character Size*
0	0	0	5-*bit*
0	0	1	6-*bit*
0	1	0	7-*bit*
0	1	1	8-*bit*
1	0	0	*Reserved*
1	0	1	*Reserved*
1	1	0	*Reserved*
1	1	1	9-*bit*

Sumber: ATMEL (2011)

f. Bit 0 - *Clock Polarity*: bit yang hanya dapat digunakan pada mode *synchronous*. Bit ini berhubungan dengan perubahan data keluaran dan sampel masukan, dan *clock synchronous* (XCK).

E. USART *Baud Rate Register*

Tabel 2.18 USART *Baud Register* ATmega32

Bit	15	14	13	12	11	10	9	8	
	URSEL	-	-	-	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Sumber: ATMEL (2011)

Keterangan:

a. Bit 15 - *Register Select*: bit ini digunakan untuk memilih akses register UCSRC (dengan nilai satu) atau register UBRRH (dengan nilai nol) dikarenakan UCSRC dan UBRRH berbagi lokasi yang sama pada register I/O
b. Bit 14:12 - *Reserved Bits*: bit-bit yang tidak digunakan dan harus di-*clear* ketika melakukan penulisan pada register UBRRH
c. Bit 11:0 - USART *Baud Rate Register*: bit-bit yang digunakan untuk menentukan nilai *baud rate* USART sesuai dengan persamaan yang ada pada Tabel 2.11.

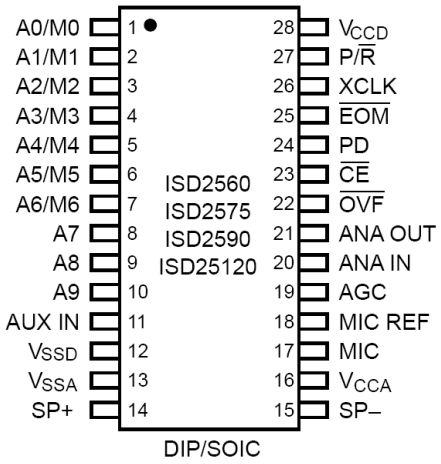
2.2. ISD25120

ISD25120 adalah IC yang berfungsi untuk merekam suara dan melakukan *playback* hasil rekaman. Durasi penyimpanan suara dari ISD25120 adalah 120 detik. Suara yang telah direkam akan disimpan kedalam alamat sebanyak 8-bit. Pengaksesan alamat dilakukan melalui pin 1 - pin 8. Karena ISD25120 memiliki alamat sebanyak 8-bit dan durasi penyimpanan suara selama 120 detik, maka tiap alamat dapat menampung durasi suara selama $120 / 2^8 = 120$

/ 256 = 0.46875 detik.

ISD25120 difungsikan sebagai antarmuka *user* pada alat yang akan dibuat. Adapun fitur-fitur yang dimiliki adalah sebagai berikut (Information Storage Device, 2000):

a. Penggunaan *chip* yang mudah untuk *record* ataupun *playback* hasil rekaman
b. Hasil suara *playback* yang berkualitas tinggi
c. Dapat dikendalikan melalui *microcontroller* ataupun saklar
d. Mempunyai durasi rekaman selama 120 detik
e. Hanya membutuhkan arus sebesar 1 µA saat mode *standby*
f. Memori yang bersifat *Non-volatile* untuk penyimpanan suara
g. Dapat diakses dengan metode *address*
h. Lama penyimpanan data mencapai 100 tahun
i. Ketahanan *record* mencapai 100.000 kali
j. Terdapat *clock* internal
k. Tidak memerlukan pemrograman pada *chip* untuk penggunaanya
l. Tegangan catu daya +5 V



Gambar 2.14 Konfigurasi pin ISD25120 (Information Storage Device, 2000)

Deskripsi pin ISD25120 yang terdapat pada Gambar 2.14 secara umum adalah sebagai berikut (Information Storage Device, 2000):

a. VCCA, VCCD : Sumber tegangan +5V DC (pin 16 dan pin 28)

b. VSSA, VSSD : Pin yang dihubungkan dengan *ground* sebagai referensi untuk VCCA dan VCCD (pin 13 dan pin 12)

c. PD : Pada saat tidak merekam atau melakukan *playback*, pin PD harus di-*set* agar konsumsi daya ISD25120 rendah. Memberikan logika *high* pada PD akan me-*reset pointer* alamat ke nilai awal (pin 24)

d. $\overline{CE}$: Pin ini harus diberi pulsa *low* minimal selama 100 nS untuk mengijinkan operasi *record* dan *playback*

e. P/$\overline{R}$: Kondisi logika dari pin P/$\overline{R}$ akan di-*latch* pada saat pin $\overline{CE}$ berlogika *low*. Logika *high* pada pin ini akan mengaktifkan operasi *playback* sedangkan logika *low* pada pin ini akan mengaktifkan operasi *record* (pin 27)

f. $\overline{EOM}$: Suatu penanda yang bersifat *non-volatile* akan disisipkan pada akhir setiap pesan yang direkam. Penanda tersebut tidak akan hilang sampai pada alamat pesan yang dimaksud dilakukan operasi *record* kembali. Pin $\overline{EOM}$ akan memberikan keluaran pulsa *low* selama 25 mS pada akhir setiap pesan yang di-*playback* (pin 25)

g. $\overline{OVF}$: Pin ini akan memberika pulsa *low* selama 6.5 µS saat *pointer* sudah berada di alamat terakhir, yaitu sudah terisi penuh (pin 22)

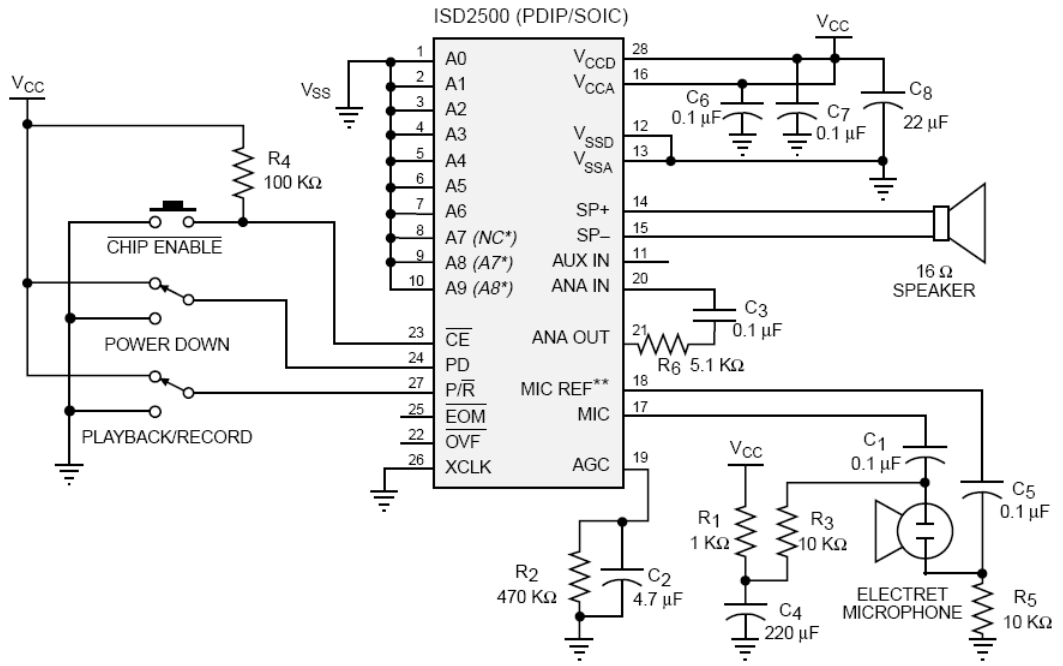
h. MIC : Pin yang digunakan untuk menyalurkan sinyal masukan ke *preamplifier* ISD25120 (pin 17)

i. MICREF : Pin yang digunakan untuk masukan *inverting* pada *microphone amplifier* (pin 18)

j. AGC : AGC akan secara otomatis mengatur *gain* yang diperlukan untuk suara yang direkam. Dengan ini maka ISD25120 dapat mendeteksi dari suara bisikan sampai suara yang keras. Untuk penggunaannya, pin AGC akan dihubungkan dengan resistor dan kapasitor secara paralel seperti yang terlihat pada Gambar 2.15 (pin 19)

k. ANA OUT : Pin ini memberikan keluaran dari *preamplifier* kepada *user* (pin 21)

l. ANA IN : Pin ini menyalurkan sinyal analog yang diterimanya untuk proses *record*. Jika sinyal yang ingin direkam berasal dari *microphone*, maka pin ini dikonfigurasikan seperti yang terlihat pada Gambar 2.15, namun jika sinyal yang ingin direkam berasal dari sumber selain *microphone*, maka sinyal tersebut dapat dihubungkan langsung dengan pin ANA IN melalui kapasitor (pin 20)



Gambar 2.15 Rangkaian ISD25120 yang dikontrol menggunakan saklar (Information Storage Device, 2000)

m. XCLK : Pin untuk masukan *clock* eksternal yang digunakan untuk menentukan *sampling* dari ISD25120. Apabila tidak digunakan pin ini harus dihubungkan dengan *ground* (pin 26)

n. SP+/SP- : Pin untuk keluaran *speaker* (pin 14 dan pin 15)

o. AUX IN : Pin yang digunakan untuk menghubungkan sinyal *playback* dari *chip* selanjutnya ke *speaker driver* chip sebelumnya jika *cascade* terhadap ISD dilakukan (pin 11)

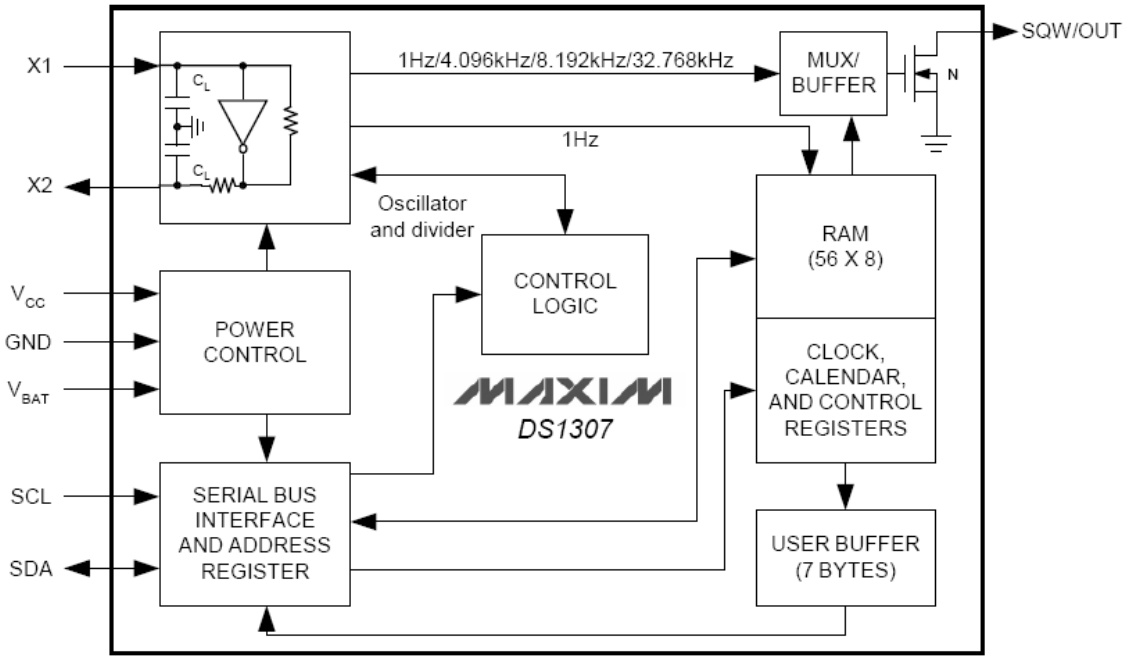
p. Ax/Mx : Pin yang digunakan sebagai masukan alamat (jika pin A8 atau pin A9 berlogika *low*) atau masukan mode operasi (jika pin A8 dan pin A9 berlogika *high*). Nilai dari Ax/Mx akan di-*latch* saat pin $\overline{CE}$ berlogika *low*.

2.3. RTC DS1307

DS1307 adalah RTC serial dengan konsumsi daya rendah yang menggunakan format BCD (*binary-coded decimal*) untuk waktu dan penanggalannya. Pengaksesan DS1307 dilakukan secara serial melalui jalur I^2C. Informasi yang dapat diberikan DS1307 adalah detik, menit, jam, hari, tanggal, bulan, dan tahun. Jumlah hari dalam satu bulan sudah diperhitungkan secara otomatis dari RTC. Terdapat dua format untuk informasi waktu yaitu format 24 jam atau format 12 jam dengan indikator AM/PM. Pada DS1307 terdapat rangkaian pendeteksi catu daya yang secara otomatis mengganti sumber catu daya-nya ke baterai 3 V apabila sumber catu daya utama dinonaktifkan. Diagram blok untuk DS1307 diilustrasikan pada Gambar 2.16. (MAXIM, 2008)

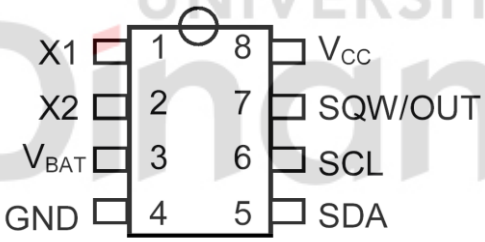
Fitur-fitur dari DS1307 adalah sebagai berikut (MAXIM, 2008):

a. Menyediakan informasi dan melakukan perhitungan detik, menit, jam, hari, tanggal, bulan, dan tahun (termasuk tahun kabisat) yang valid sampai dengan tahun 2100
b. RAM untuk penyimpanan data berkapasitas 56 Byte yang ditunjang oleh baterai
c. Diakses menggunakan komunikasi serial I^2C
d. Sinyal keluaran SQW (*Square-Wave*) yang dapat diatur melalui program
e. Secara otomatis mengganti sumber catu daya-nya ke baterai saat catu daya utama mati
f. Membutuhkan arus kurang dari 500 nA saat bekerja menggunakan baterai



Gambar 2.16 Diagram blok DS1307 (MAXIM, 2008)

2.3.1. Konfigurasi Pin DS1307



Gambar 2.17 Konfigurasi pin RTC DS1307 (MAXIM, 2008)

Deskripsi pin DS1307 yang terdapat pada Gambar 2.17 secara umum adalah sebagai berikut (MAXIM, 2008):

a. X1 dan X2 : Pin-pin yang akan dihubungkan pada kristal dengan nilai frekuensi 32.768kHz (pin 1 dan pin 2)

b. V_{BAT} : Catu daya cadangan yang terhubung pada baterai atau sumber daya cadangan lainnya dangan standar tegangan 3 V (pin 3)

c. GND : Ground (pin 4)

d. SDA : SDA (*Serial Data Input/Output*) adalah jalur data masukan maupun keluaran untuk antarmuka serial I^2C. Pin SDA memerlukan resistor *pull-up* (pin 5)

e. SCL : SCL (*Serial Clock Input*) adalah jalur masukan bagi sinyal *clock* yang dikeluarkan oleh *master device* untuk sinkronisasi data dalam antarmuka serial I^2C. Pin ini memerlukan resistor *pull-up* (pin 6)

f. SQW/OUT : *Square-Wave Output* ini bila bit SQWE di-*set* akan menghasilkan sinyal keluaran dengan frekuensi tertentu (1 Hz, 4 kHz, 8 kHz, atau 32 kHz). Pin ini memerlukan pull-up eksternal (pin 7)

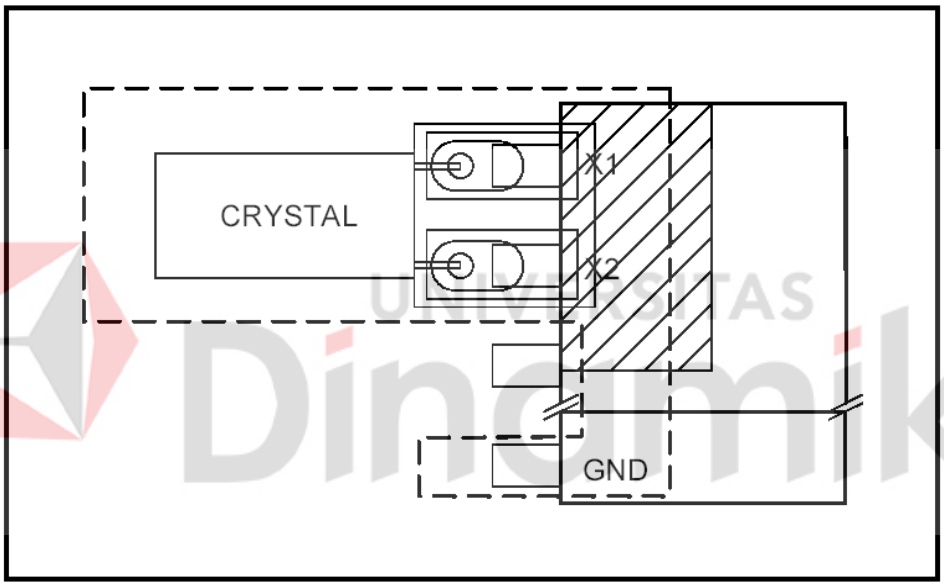
g. V_{CC} : Pin untuk masukan catu daya utama (pin 8).

DS1307 beroperasi sebagai *slave device* pada proses komunikasi serial I^2C. Proses pengaksesan dimulai dengan memberikan kondisi start dan nomor ID (*Identification*) dari DS1307 yang dilanjutkan dengan alamat register mana yang akan dilakukan operasi baca atau tulis. Proses baca/tulis dapat dilakukan pada register yang telah dipilih sampai kondisi stop diberikan.

Bila nilai tegangan dari catu daya utama kurang dari 1.25 kali nilai tegangan pada V_{BAT}, DS1307 akan menghentikan proses pengaksesan dan me-*reset* pencacah alamat register. Dalam kondisi ini perangkat tidak dapat mengenali masukan dan tidak dapat diakses untuk mencegah kesalahan dalam komunikasi. Bila nilai tegangan catu daya utama kurang dari nilai tegangan pada V_{BAT}, perangkat akan mengganti sumber catu daya dari catu daya utama ke catu daya cadangan (baterai).

2.3.2. Sirkuit Osilator

DS1307 menggunakan kristal eksternal sebesar 32,768 kHz namun tidak memerlukan resistor maupun kapasitor eksternal karena di dalam IC DS1307 sudah terdapat kapasitor untuk rangkaian osilator. Kristal sebaiknya dipasang dengan dikelilingi oleh jalur ground yang tersambung dengan pin GND untuk mengurangi *noise*, seperti pada Gambar 2.18. Untuk mengaktifkan osilator, pada saat inisialisasi alamat register 00h bit ke-7 (CH) harus diberi nilai 0.



Gambar 2.18 *Layout* yang disarankan untuk peletakan kristal (MAXIM, 2008)

2.3.3. Peta Alamat RTC dan RAM

Tabel 2.19 adalah tabel peta alamat register RTC dan RAM dari DS1307. Register RTC beralamat di 00h sampai dengan 07h. Sedangkan sisanya, 08h sampai 3Fh merupakan alamat untuk register RAM. Dalam pengaksesan secara sekuensial, ketika *pointer* alamat mencapai alamat 3Fh, alamat selanjutnya adalah alamat awal, yaitu 00h.

Tabel 2.19 Peta alamat DS1307

Alamat	Bit								Fungsi	Rentang
	7	6	5	4	3	2	1	0		
00h	CH	Detik x10			Detik				Detik	00 – 59
01h	0	Menit x10			Menit				Menit	00 – 59
02h	0	12	Jam x10		Jam				Jam	01 – 12 +AM/PM 00 – 23
		24	AM/PM	Jam x10						
03h	0	0	0	0	0	Hari			Hari	01 – 07
04h	0	0	Tanggal x10		Tanggal				Tanggal	01 – 31
05h	0	0	0	Bulan x10	Bulan				Bulan	01 – 12
06h	Tahun x10				Tahun				Tahun	00 – 99
07h	OUT	0	0	SQWE	0	0	RS1	RS2	Control	–
08h – 3Fh									RAM 56 x 8	00h – FFh

Sumber: MAXIM (2008)

2.3.4. Register Kontrol (*Control Register*)

Register kontrol (07h) pada DS1307 digunakan untuk mengendalikan operasi dari pin SQW/OUT dengan detail:

a. Bit 7 : *Output Control* (OUT). Bit ini digunakan untuk mengendalikan level keluaran pada pin SQW/OUT saat *Square-Wave Output* tidak diaktifkan. Bila bit SQWE = 0, pin SQW/OUT akan berlogika *high* jika bit OUT = 1 dan akan berlogika *low* bila bit OUT = 0

b. Bit 4 : *Square Wave Enable* (SQWE). Bila bit ini berlogika *high*, keluaran osilator akan aktif. Frekuensi *Square-Wave Output* ditentukan dari nilai bit RS0 dan RS1

c. Bit 1 & Bit 0 : Kedua bit ini mengendalikan frekuensi *Square-Wave Output* ketika diaktifkan. Pemilihan frekuensi berdasarkan bit RS dapat dilihat pada Tabel 2.20.

Tabel 2.20 Frekuensi *Square-Wave Output*

RS1	RS0	Frekuensi *Square-Wave Output*
0	0	1 Hz
0	1	4,096 kHz
1	0	8,192 kHz
0	0	32,768 kHz

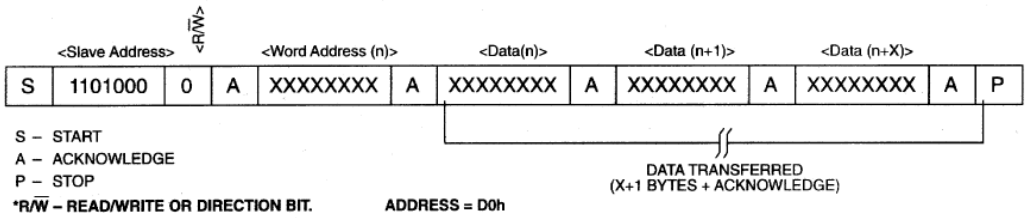
Sumber: MAXIM (2008)

2.3.5. Mode Operasi DS1307

DS1307 memiliki 2 mode operasi dalam proses transfer data menggunakan I^2C, yaitu:

A. *Slave Receiver Mode* (DS1307 dalam mode tulis)

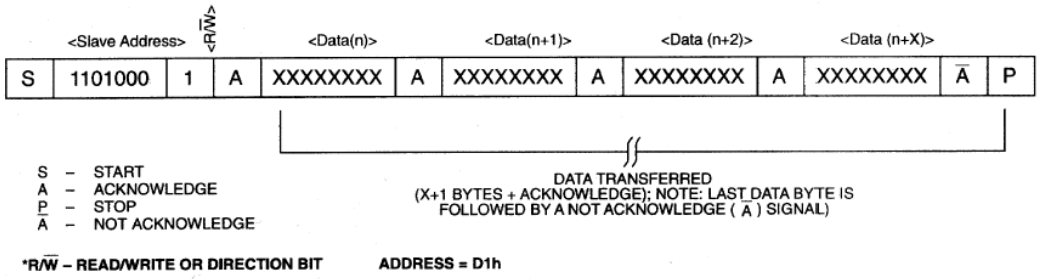
Data serial dan *clock* diterima melalui pin SDA dan SCL. Setelah setiap Byte diterima, sebuah bit *acknowledge* (ACK) akan dikirimkan. Kondisi start dan stop digunakan untuk mengawali dan mengakhiri proses transfer data. Byte alamat adalah Byte pertama yang diterima setelah sinyal start diberikan oleh *master*. Byte address berisi 7 bit alamat DS1307, 1101000b, diikuti oleh 1 bit direksi transfer (R/W). Untuk menulis kepada DS1307, bit ini diberi nilai 0. Setelah menerima byte alamat yang sesuai DS1307 akan mengirimkan sinyal ACK pada *master* melalui jalur SCL. Setelah mengkonfirmasi alamat dan bit direksi, *master* akan mengirimkan alamat register yang akan diakses pada DS1307. Hal ini akan menyebabkan *pointer* alamat berada pada alamat yang ditunjuk tersebut. Setelah itu *master* akan mulai mengirimkan data yang harus ditulis pada register sesuai dengan data yang ditunjuk oleh *pointer* alamat secara per Byte dengan dipisahkan oleh sinyal ACK. Setelah selesai, *master* akan mengirim sinyal stop pada DS1307 untuk menghentikan proses penulisan data. Ilustrasi *Slave Receiver Mode* untuk DS1307 terdapat pada Gambar 2.19.



Gambar 2.19 *Slave Receiver Mode* DS1307 (DS1307 dalam mode baca) (MAXIM, 2008)

B. *Slave Transmitter Mode* (DS1307 dalam mode baca)

Sinyal start dan Byte pertama diproses seperti dalam mode penerima. Perbedaan pada mode pengirim ini bit direksi bernilai kebalikannya. Byte pertama berisi 7 bit alamat DS1307, yaitu 1101000b, diikuti oleh bit direksi yang bernilai 1 untuk proses membaca dari DS1307 (baca). Setelah menerima Byte alamat yang sesuai DS1307 akan mengirimkan sinyal ACK pada *master* melalui jalur SCL. Setelah mengkonfirmasi alamat dan bit direksi, *master* akan mengirimkan alamat register yang akan diakses pada DS1307. Hal ini akan menyebabkan *pointer* alamat berada pada alamat yang ditunjuk tersebut. Bila master tidak mengirimkan alamat register yang akan diakses, pointer DS1307 akan berada di alamat register terakhir. Data serial dikirimkan oleh DS1307 melalui jalur SDA berdasarkan serial *clock* yang diterima sebagai masukan pada jalur SCL. Setiap Byte data yang dikirim harus dipisahkan sinyal ACK dari *master*. Untuk mengakhiri proses pembacaan, DS1307 harus menerima sinyal *Not Acknowledge* dan stop. Ilustrasi untuk proses ini dapat dilihat pada Gambar 2.20.



Gambar 2.20 *Slave Transmitter Mode* DS1307 (DS1307 dalam mode tulis) (MAXIM, 2008)

2.4. Penentuan Waktu Sholat Lima Waktu

Penentuan waktu sholat mula-mula dilakukan dengan Rukyat (observasi/pengamatan) posisi matahari. Namun dengan kemajuan ilmu pengetahuan, tanpa melihat posisi matahari, manusia dapat mengetahui kapan datangnya waktu sholat, yaitu melalui perhitungan.

2.4.1. Penentuan Waktu Sholat dengan Rukyat

1. Dhuhur

Waktu Dhuhur dimulai saat pertengahan hari, yaitu ketika matahari melewati garis meridian (lingkaran besar langit yang menghubungkan utara dan selatan). Saat melewati garis meridian, ada tiga kemungkinan *azimuth* matahari (dihitung dari arah utara). Pertama, *azimuth* matahari = 0 derajat, yaitu ketika matahari melewati garis meridian, posisinya di belahan langit utara. Kedua, *azimuth* = 180 derajat, ketika posisinya di belahan langit selatan. Ketiga, *azimuth*-nya tidak dapat ditentukan, ketika posisinya benar-benar tepat di *zenith* (atas kepala) atau ketinggiannya tepat 90 derajat.

Untuk kemungkinan pertama dan kedua, sebuah benda memiliki panjang bayangan jika terkena sinar matahari. Adapun untuk kemungkinan ketiga, panjang bayangan sama dengan nol. Panjang bayangan saat datangnya waktu Dhuhur ini

akan berpengaruh pula pada penentuan datangnya waktu sholat Ashar. Waktu Dhuhur berakhir saat datangnya waktu sholat Ashar. Waktu-waktu sholat lainnya akan ditentukan berdasarkan waktu sholat Dhuhur. (Anugraha, 2009c)

2. Ashar

Ada dua pendapat mengenai kapan datangnya waktu sholat Ashar. Ini berkaitan dengan bayangan suatu benda yang ditegakkan di atas tanah (seperti pohon dan tiang). Menurut *mazhab* Syafii, waktu shalat Ashar adalah ketika panjang bayangan suatu benda sama dengan tinggi benda tersebut (ditambah panjang bayangan saat Dhuhur). Sedangkan menurut *mazhab* Hanafi, waktu sholat Ashar adalah ketika panjang bayangan suatu benda sama dengan dua kali tinggi benda tersebut (ditambah panjang bayangan saat Dhuhur).

Jika bayangan benda saat Ashar = Sa, bayangan benda saat Dhuhur = Sd dan tinggi benda = h, maka secara sederhana dapat ditulis Sa = h + Sd menurut mazhab Syafii dan Sa = 2*h + Sd menurut mazhab Hanafi. Waktu Ashar berakhir saat datangnya waktu sholat Maghrib. (Anugraha, 2009c)

3. Maghrib

Waktu sholat Maghrib dimulai saat matahari terbenam. Ketika matahari terbenam dimana posisinya di bawah ufuk, langit tidak langsung gelap. Hal ini disebabkan adanya atmosfer bumi yang membiaskan cahaya matahari. Karena itu, matahari harus tenggelam hingga belasan derajat di bawah ufuk supaya tidak ada lagi cahaya matahari yang dapat dibiaskan sehingga langit menjadi gelap. Waktu sholat Maghrib berakhir saat datangnya waktu sholat Isya'. (Anugraha, 2009c)

4. Isya’

Waktu sholat Isya' dimulai saat langit gelap, atau berakhirnya mega merah (*astronomical twilight*) di langit barat. Waktu Isya' berakhir saat datangnya waktu Subuh. (Anugraha, 2009c)

5. Subuh

Waktu Subuh dimulai ketika munculnya fajar atau cahaya secara merata di langit timur. Meskipun saat itu matahari masih belasan derajat di bawah ufuk, namun akibat pembiasan atmosfer cahaya matahari dapat dibiaskan sehingga langit tidak lagi gelap. Waktu Subuh berakhir saat matahari terbit. (Anugraha, 2009c)

Ada perbedaan pendapat untuk menentukan datangnya waktu sholat Isya' maupun Subuh. Penentuan kedua waktu tersebut tidak secara langsung berkaitan dengan posisi matahari, namun efek dari atmosfer yang membiaskan cahaya matahari dari bawah ufuk. Ada beberapa pendapat, misalnya *altitude* (ketinggian) matahari itu berkisar antara 15 hingga 20 derajat (disebut juga sudut *twilight*) di bawah ufuk agar tidak ada lagi cahaya matahari yang dapat dibiaskan. Tidak ada pendapat yang pasti mengenai sudut ini, sehingga perbedaan satu derajat saja akan berpengaruh pada perbedaan waktu sholat Isya' dan Subuh beberapa menit. Karena perhitungan waktu sholat akan dibandingkan dengan jadwal sholat pada situs PKPU (Pos Keadilan Peduli Ummat), maka sudut *twilight* yang digunakan disesuaikan dengan sudut *twilight* yang digunakan oleh PKPU yaitu, 18° untuk sholat Isya’ dan 20° untuk sholat Subuh. (Anugraha 2009c)

2.4.2. Penentuan Waktu Sholat dengan Ilmu Hisab

Secara bahasa, kata "hisab" berasal dari *haasaba – yuhaasibu – muhaasabatan – hisaaban*. Kata hisab berarti perhitungan, sedangkan ilmu hisab sendiri adalah suatu ilmu yang digunakan untuk melakukan perhitungan posisi benda langit. Ilmu ini memiliki kaitan erat dengan astronomi, namun secara umum ilmu hisab hanya mengambil bagian kecil dari astronomi yaitu mempelajari pergerakan matahari, bulan, bumi serta planet-planet lain di tata surya (*solar system*). Dengan mempelajari ilmu hisab, kita akan dapat menentukan arah qiblat, waktu sholat, serta posisi matahari dan bulan setiap saat. Selain itu, kalender Islam dapat pula dihitung, sehingga masuknya bulan-bulan penting dalam Islam seperti Muharram, Ramadhan, Syawal dan Dzulhijjah dapat diperkirakan. Dengan ilmu hisab, berbagai peristiwa alam yang menakjubkan seperti gerhana matahari, gerhana bulan, transit Merkurius dan Venus di matahari dapat pula dihitung dengan akurasi tinggi. Dan masih banyak lagi fenomena yang dapat ditelusuri melalui ilmu hisab. (Anugraha, 2009b)

Perhitungan kelima waktu sholat dilakukan dengan beberapa informasi antara lain tanggal, letak geografis (lintang, bujur, zona waktu, dan ketinggian) dari tempat yang ingin diketahui jadwal sholat-nya, *equation of time*, dan deklinasi matahari.

1. J2000

Menurut Wicaksono (2004), untuk menghindari kerumitan perhitungan astronomi yang melibatkan kalender, digunakan sistem "penomoran hari". Penggunaan J2000 banyak digunakan untuk mengindekskan sistem penanggalan. Huruf "J" merupakan kepanjangan dari "Julian". Sebagai acuan, pada tanggal 1

Januari 2000, nilai J2000 adalah sama dengan 0. demikian tiap satu hari, nilai J2000 bertambah sebesar 1 satuan.

Nilai J2000 bertambah satu satuan tiap pukul 12:00 siang (bukan tiap jam 00:00). Karena tiap satu hari nilai J2000 bertambah satu satuan, padahal dalam sehari ada 24 jam, maka setiap jam nilai J2000 bertambah 1/24 satuan. Misalnya, nilai J2000 pada tanggal 3 Januari 2000 jam 12:00 adalah 2, maka nilai J2000 pada tanggal 3 Januari 2000 pukul 14:00 adalah 2 + (2 / 24) = 2.08333. contoh lain, nilai J2000 pada tanggal 24 Februari 2004 jam 12:00 adalah 1515, maka nilai J2000 pada tanggal 24 Februari 2004 jam 6:00 adalah 1515 - (6 / 24) = 1514.75.

Nilai J2000 sebelum tanggal 1 Januari 2000 jam 12:00 bernilai negatif. Nilai J2000 adalah untuk waktu UTC, jadi dimanapun tempat di dunia ini memiliki nilai J2000 yang sama pada waktu yang sama. Misalnya, nilai J2000 di Surabaya pada tanggal 1 Februari 2000 jam 21:00 WIB, maka harus mengoreksi ke UTC (yakni WIB -7), atau 1 Februari 2000 jam 14:00 UTC, yakni sebesar 31.08333.

2. Letak Geografis

Untuk menentukan suatu lokasi atau daerah di suatu belahan bumi, maka diperlukan garis-garis vertikal yang menghubungkan kutub utara dan kutub selatan yang dinamakan garis bujur dan garis-garis horizontal yang mengelilingi bumi yang disebut dengan garis lintang. Selain itu, terdapat garis lintang yang membelah bumi menjadi belahan bumi utara dan belahan bumi selatan yang disebut dengan garis khatulistiwa. Garis-garis yang terletak di sebalah utara garis khatulistiwa dinamakan garis Lintang Utara (LU), sedangkan garis-garis yang terletak di sebelah selatan garis khatulistiwa dinamakan garis Lintang Selatan

(LS). Masing-masing garis lintang (LU dan LS) memiliki rentang dari 0° hingga 90°, sedangkan garis lintang 0° merupakan garis khatulistiwa. Garis bujur adalah garis dari suatu tempat yang diukur dari titik garis kota Greenwich London Inggris ke arah timur dan ke arah barat. Kota Greenwich sebagai dasar titik garis pengukurannya yang mempunyai nilai 0°. Jika diukur dari titik garis kota Greenwich 0° ke arah timur sampai 180° disebut Bujur Timur (BT), sedangkan jika diukur dari titik garis kota Greenwich 0° ke arah barat sampai 180° disebut Bujur Barat (BB).

Ada dua macam penulisan garis lintang maupun garis bujur, yaitu pertama dengan menyertakan arahnya setelah besarnya derajat (contoh : 6° LU, 11° LS, 15° BT, 7° BB). Kedua, dengan tidak menyertakan arahnya, tetapi dengan menggunakan tanda +/-. Yaitu tanda + (atau bisa juga dengan tidak menuliskan tanda positif tersebut) untuk nilai Lintang Utara dan nilai Bujur Timur, sedangkan tanda - (negatif) untuk nilai Lintang Selatan dan nilai Bujur Barat (contoh : 6°, -11°, 15°, -7°). (Wicaksono, 2004)

3. Algoritma Konversi di Bidang Busur

Seringkali dijumpai satuan dalam menunjukan besarnya sudut dalam bentuk desimal ataupun busur menit (*Minutes of Arc*). Bentuk *Minutes of Arc* ini sering kali dijumpai pada penentuan posisi geografis. Misalkan, lokasi rumah berada di 7°48’32.89”s (dibaca tujuh derajat, empat puluh delapan menit, tiga dua koma delapan sembilan detik, lintang selatan). Bentuk busur tersebut lebih mudah dibaca manusia (karena terpisah-pisah) daripada bentuk desimal. Berbeda jika ingin melakukan kalkulasi dengan nilai masukan berdasar pada satuan busur

seperti tadi. Misalkan untuk nilai 8.5° dalam bentuk busur akan ditulis 8°30”. (Faithtear, 2007)

Berikut ini adalah jabaran algoritma untuk melakukan konversi dari satuan bentuk busur ke bentuk desimal.

Dari format busur ke desimal

dd°mm’ss.ss”

ss.ss dibagi dengan 60, ss.ss / 60 ..... (2.1)

Tambahkan dengan nilai mm, mm + (ss.ss / 60), menjadi format

dd°mm.mmm’ ..... (2.2)

mm.mmm bagikan dengan 60, mm.mmm / 60..... (2.3)

Tambahkan dengan nilai dd, menjadi format dd.ddddd..... (2.4)

Contoh :

7°48’32.89” akan dikonversikan ke dalam bentuk desimal

a. Pada bagian detik dibagi 60

   32.89 / 60 = 0.548166667

b. Tambahkan nilai menit ke hasil tadi

   48 + 0.548166667 = 48.548166667

c. Nilai menit yang desimal dibagi 60

   48.548166667 / 60 = 0.809136111

d. Tambahkan nilai derajat ke hasil tadi

   7 + 0.809136111

e. Telah didapat nilai konversinya dalam bentuk desimal ialah 7,809136111°.

dd.ddddd = dd + ((mm + (ss.ss / 60)) / 60)

Dari format desimal ke busur

dd.ddddd°

dd.ddddd dikurangi dengan dd ..... (2.5)

0.ddddd kalikan dengan 60 = mm.ddd..... (2.6)

Kurangi mm.ddddd dengan mm ..... (2.7)

0.ddd kalikan dengan 60 = ss.d ..... (2.8)

Contoh :

110.349873493° akan dikonversikan kedalam bentuk busur

a. Nilai awal di-*modulus* (pembagian dengan sisa baginya) dengan 1 sedangkan hasil *divide* (pembagian bulat) masuk dalam bagian derajat

   110.349873493 mod 1 = 0.349873493

   dan 110.349873493 div 1 = 110

b. Nilai hasil *modulus* tadi dikalikan dengan 60

   0.349873493 * 60 = 20.99240958

c. Nilai hasil perkalian tadi di-*modulus* dengan 1 sedangkan hasil div masuk ke dalam bagian menit

   20.99240958 mod 1 = 0.99240958

   dan 20.99240958 div 1 = 20

d. Nilai hasil *modulus* tadi dikalikan dengan 60

   0.99240958 * 60 = 59.5445748

e. Hasil konversi nilai dalam bentuk busur ialah 110°20’59.5446”

dd = dd.ddddd div 1

mm = ((dd.ddddd mod 1) * 60) div 1

ss.ss = (((dd.ddddd mod 1) * 60) mod 1) * 60

4. Zona Waktu

Indonesia terbagi atas 3 daerah waktu, yaitu Waktu Indonesia Barat (WIB), Waktu Indonesia Tengah (WITA) dan Waktu Indonesia Timur (WIT). Sedangkan di dunia terbagi atas 24 zona waktu yang berbeda. Sebagai waktu standar adalah waktu *Greenwich Mean Time* (GMT) atau sejak tahun 1986 telah diubah menjadi *Universal Time Coordinate* atau disingkat UTC. WIB memiliki zona waktu UTC +7, WITA memiliki zona waktu UTC +8 dan WIT memiliki zona waktu UTC +9.

5. *Equation of Time*

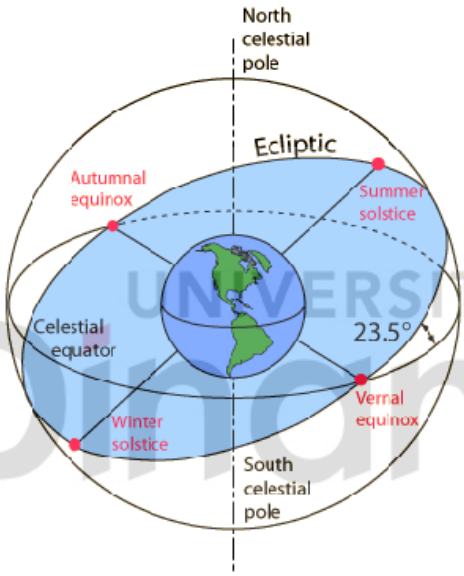
Matahari tidak selalu mencapai titik tertinggi pada waktu yang sama setiap harinya. Namun terdapat selisih ± 15 menit agar matahari benar-benar mencapai titik tertingginya. Karena penentuan waktu sholat dilakukan berdasarkan titik tertinggi matahari, maka perlu dilakukan perhitungan selisih antara waktu matahari mencapai titik tertinggi rata-rata dengan waktu matahari mencapai titik tertinggi sesungguhnya. Selisih ini disebut dengan *equation of time*.

Karena standar waktu internasional adalah kota Greenwich dengan busur yang bernilai tepat 0°, maka waktu matahari mencapai titik tertinggi rata-rata adalah pukul 12:00:00 waktu setempat. Jika matahari mencapai titik tertinggi sesungguhnya adalah pukul 12:02:42, maka *equation of time*-nya adalah *minus* 02:42 (karena terlambat). (Anugraha, 2010)

6. Deklinasi Matahari

Matahari tidak sepanjang waktu beredar di khatulistiwa, ini disebabkan oleh sumbu bumi miring terhadap bidang ekliptika dalam berevolusi mengelilingi

matahari. Hal ini menyebabkan matahari mengalami pergeseran ke utara dan selatan. Pergeseran tersebut terjadi antara Garis Balik Utara (GBU) dan Garis Balik Selatan (GBS). Daerah antara GBU dan GBS disebut dengan daerah tropis. Deklinasi adalah sudut antara lintasan semu harian benda langit dengan ekuator langit. Sudut ini bernilai positif jika ke utara dan bernilai negatif jika ke selatan. Besarnya deklinasi matahari antara -23.5° hingga +23.5°, seperti yang ditunjukkan pada Gambar 2.21. (Edwards, 2007)



Gambar 2.21 Deklinasi matahari (Edwards, 2007)

7. Rumus Perhitungan Waktu Sholat

Untuk menghitung waktu sholat pada daerah dan waktu tertentu diperlukan informasi letak lintang, letak bujur, zona waktu, dan ketinggian dari daerah tersebut serta tanggal yang hendak dicari.

Menurut Britain (1995), persamaan 2.9 sampai persamaan 2.17 dapat digunakan untuk menghitung deklinasi matahari dan *equation of time* pada tanggal tertentu. Sebagai masukan (*input)* adalah tanggal yang telah diubah kebentuk J2000, yaitu variabel k.

$$L = 280.461 + (0.9856474 * k) \quad \text{(2.9)}$$

$$g = 357.528 + (0.9856003 * k) \quad \text{(2.10)}$$

$$lambda = L + (1.915 * \sin(g)) + (0.020 * \sin(2 * g)) \quad \text{(2.11)}$$

$$epsilon = 23.439 - (0.0000004 * k) \quad \text{(2.12)}$$

$$Y = \cos(epsilon) * \sin(lambda) \quad \text{(2.13)}$$

$$X = \cos(lambda) \quad \text{(2.14)}$$

$$alpha = \arctan\left(\frac{Y}{X}\right) \quad \text{(2.15)}$$

Jika $X < 0$ maka $alpha = alpha + 180$, jika $Y < 0$ dan $X > 0$ maka

$alpha = alpha + 360$, selain itu $alpha = alpha$.

$$delta = \arcsin(\sin(epsilon) * \sin(lambda)) \quad \text{(2.16)}$$

$$et = (L - alpha) * 4 \quad \text{(2.17)}$$

Keterangan:

k = J2000

L = Bujur rata-rata matahari

g = Anomali rata-rata matahari

lambda = *The ecliptic longitude of the sun*

epsilon = kemiringan orbit rata-rata bumi

X, Y = digunakan untuk menentukan nilai sudut *alpha*

alpha = *The right ascension of the sun*

delta = *The declination of the sun*

et = *Equation of time*

Yang dimasksud dengan bujur rata-rata matahari (L) adalah nilai bujur yang dapat ditentukan apabila orbit dari matahari berbentuk lingkaran, bebas dari *perturbation* (pengaruh gravitasi dari benda planet lain), dan mempunyai sudut inklinasi nol. Yang dimaksud dengan anomaly rata-rata matahari (g) adalah parameter posisi dan waktu suatu benda yang bergerak di orbit kepler, dimana anomali didasarkan pada hukum kepler II yaitu: "Luas daerah yang disapu oleh garis antara matahari dengan planet adalah sama untuk setiap periode waktu yang sama".

Nilai L dan g harus berada dalam rentang 0 hingga 360. Apabila dalam perhitungan ditemukan nilai L dan g yang nilainya lebih kecil dari 0, maka harus ditambah dengan 360 atau kelipatannya, sehingga berada dalam rentang 0 - 360. Demikian pula apabila dalam perhitungan ditemukan nilai L dan g yang nilainya lebih besar dari 360, maka harus dikurangi dengan 360 atau kelipatannya, sehingga berada dalam range 0-360. Semua perhitungan trigonometri di atas menggunakan satuan derajat (bukan radian). Tanda titik (.) digunakan untuk memisahkan desimal atau sebagai pengganti tanda koma.

Setelah didapat hasil dari perhitungan deklinasi matahari dan *equation of time* pada tanggal tertentu, maka untuk menghitung waktu sholat digunakan persamaan sebagai berikut (Anugraha, 2009a):

$$Z = 12 + \frac{(TZ * 15) - LO}{15} - \frac{ET}{60} \quad \text{..... (2.18)}$$

$$U = \frac{1}{15} * \arccos\left(\frac{\sin\left(-0.8333 - \left(0.0347 * \sqrt{H}\right)\right) - \left(\sin(delta) * \sin(LA)\right)}{\cos(delta) * \cos(LA)}\right) \quad \text{..... (2.19)}$$

$$V = \frac{1}{15} * \arccos\left(\frac{-\sin(T) - (\sin(delta) * \sin(LA))}{\cos(delta) * \cos(LA)} \right) \quad \text{..... (2.20)}$$

$$W = \frac{1}{15} * \arccos\left(\frac{\sin(arc\cot(1 + \tan(|LA - delta|))) - (\sin(delta) * \sin(LA))}{\cos(delta) * \cos(LA)} \right) \quad \text{..... (2.21)}$$

$$Subuh = Z - V \quad \text{..... (2.22)}$$

$$Terbit = Z - U \quad \text{..... (2.23)}$$

$$Zawaal = Z \quad \text{..... (2.24)}$$

$$Ashar = Z + W \quad \text{..... (2.25)}$$

$$Maghrib = Z + U \quad \text{..... (2.26)}$$

$$Isya' = Z + V \quad \text{..... (2.27)}$$

Keterangan:

Z = Dhuhur

U = *Hour Angle* Terbit dan *Hour Angle* Maghrib dalam satuan jam

V = *Hour Angle* Subuh dan *Hour Angle* Isya' dalam satuan jam

W = *Hour Angle* Azhar dalam satuan jam

TZ = Zona Waktu

delta = Deklinasi Matahari

ET = *Equation of Time*

LA = Letak Lintang Tempat

LO = Letak Bujur Tempat

H = Ketinggian Tempat (meter)

T = Sudut *Twilight*

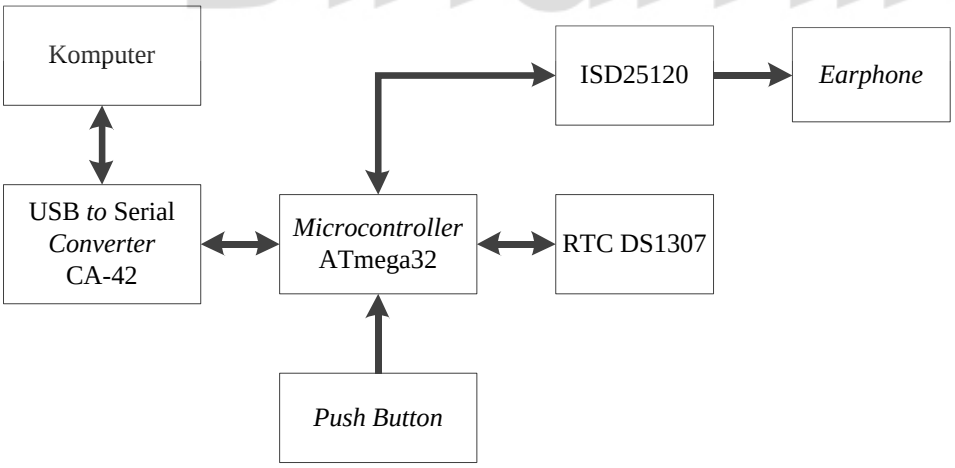
BAB III

METODE PENELITIAN

Perancangan perangkat keras dilakukan dengan metode penelitian yang didasarkan pada studi kepustakaan berupa data-data literatur dari masing-masing komponen, informasi dari *internet*, dan konsep-konsep teoretis dari buku-buku penunjang. Setelah literatur terkumpul barulah dilakukan perancangan perangkat keras yang dilanjutkan dengan perancangan dan pembuatan perangkat lunak yaitu program sederhana untuk melakukan pengujian pada tiap bagian perangkat keras yang telah dibuat dan program untuk menjalankan perangkat keras secara utuh.

3.1. Perancangan Perangkat Keras

Perancangan perangkat keras pada sistem dilakukan berdasarkan blok diagram yang terdapat pada Gambar 3.1.



Gambar 3.1 Blok diagram jam untuk tunanetra

Microcontroller yang digunakan pada sistem ini, yaitu ATmega32, akan difungsikan sebagai pengontrol sistem. Pengontrolan yang dilakukan meliputi pembacaan data waktu serta penulisan register-register pada RTC, dalam sistem

ini tipe yang digunakan adalah DS1037, memberikan instruksi *playback* pada ISD25120, mendeteksi penekanan tombol dari *user*, dan menerima serta mengirimkan data serial ke komputer yang digunakan untuk mengatur parameter-parameter yang dibutuhkan.

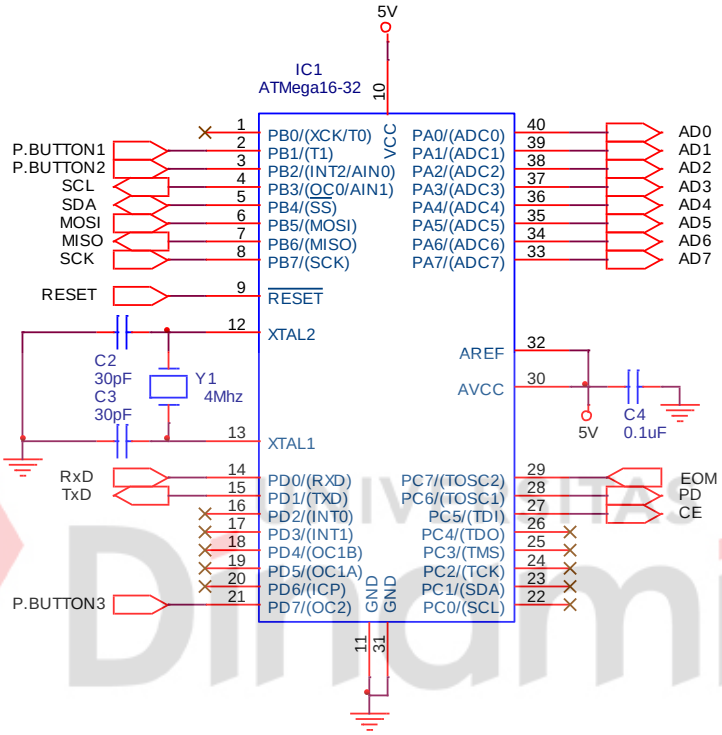
RTC digunakan sebagai acuan waktu pada sistem. Informasi yang disediakan adalah detik, menit, jam, tanggal, bulan, dan tahun. Pengaksesan RTC oleh *microcontroller* dilakukan secara serial dengan protokol komunikasi I^2C. ISD25120 berfungsi untuk memberi keluaran suara melalui *earphone* yang digunakan sebagai media informasi kepada *user* mengenai waktu aktual, peringatan waktu *sholat*, serta indikator kota pilihan *user*. Pengaksesan ISD25120 dilakukan secara paralel oleh *microcontroller*.

Agar sistem ini dapat bekerja, dibutuhkan pengaturan beberapa parameter. Pengaturan parameter dilakukan melalui komputer yang akan dihubungkan dengan *microcontroller* melalui CA-42. Komunikasi antara komputer dengan *microcontroller* dilakukan secara serial. Karena keluaran dari CA-42 kompatibel dengan *level* tegangan *microcontroller*, maka penggunaan IC seperti MAX232 tidak lagi diperlukan.

3.1.1. *Minimum System* ATmega32

Rangkaian *minimum system* ATmega32 yang digunakan pada sistem dapat diilustrasikan seperti pada Gambar 3.2. *Port* A yaitu PA0 - PA7 digunakan sebagai masukan alamat untuk instruksi *playback* pada ISD25120. Pin PC7 digunakan sebagai masukan dari pin $\overline{EOM}$ pada ISD25120 yang menandakan bahwa instruksi *playback* yang dilakukan pada alamat yang sesuai dengan *port* A sudah selesai. Pin PC6 digunakan sebagai keluaran untuk pin PD pada ISD25120.

Apabila logika *high* diberikan pada PC6, maka ISD25120 akan memasuki mode kerja dengan konsumsi daya rendah. Pin PC5 akan dihubungkan dengan pin $\overline{CE}$ pada ISD25120. Memberikan pulsa *low* selama 100 nS atau lebih pada PC5 saat PC6 berlogika *low* akan mengijinkan instruksi *playback*.



Gambar 3.2 Skema rangkaian *minimum system* ATmega32

Komunikasi serial dengan komputer dilakukan melalui pin PD0 dan pin PD1. Sedangkan pin PB1, pin PB2, dan pin PD7 digunakan untuk mendeteksi penekanan tombol dari *user*. Komunikasi dengan RTC dilakukan dengan protokol I^2C melalui pin PB3 sebagai jalur *clock* untuk sinkronisasi komunikasi dan pin PB4 sebagai jalur pertukaran data. Sedangkan pin PB5 - PB7 digunakan sebagai jalur pemrograman *microcontroller*.

Pin XTAL1 dan pin XTAL2 dihubungkan dengan kristal 4 MHz dan dua buah kapasitor dengan kapasitansi 22 pF. Pemilihan kristal dengan frekuensi 4

MHz dilakukan dengan pertimbangan agar konsumsi arus oleh *microcontroller* rendah. Sedangkan pililihan nilai kapasitansi kapasitor didasarkan pada Tabel 3.1.

Tabel 3.1 Tabel pemilihan nilai kapasitansi kapasitor

Frequency Range (MHz)	*Recommended Range for Capacitors* C1 *and* C2 *for Use with Crystals* (pF)
0.4 - 0.9	-
0.9 - 3.0	12 - 22
3.0 - 8.0	12 - 22
1.0 ≤	12 - 22

Sumber: ATMEL (2011)

Pin VCC diberi masukan tegangan operasi berkisar antara 4.5 V sampai dengan 5.5 V. Pin $\overline{RESET}$ berfungsi untuk masukan *reset* program secara otomatis atau manual. Sedangkan pin MOSI, MISO, dan SCK digunakan untuk keperluan pemrograman *microcontroller*.

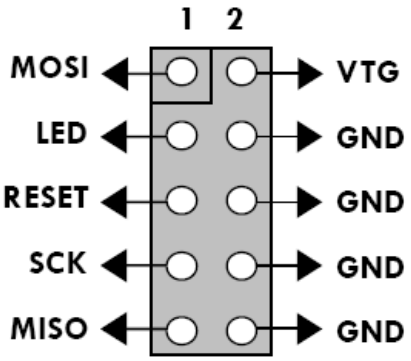
A. *Downloader*

Untuk melakukan proses *download* program, yaitu file dengan ekstensi ".hex" digunakan perangkat bantu AVR USB ISP yang akan dihubungkan dengan *port* USB (*Universal Serial Bus*) pada komputer. Sebelum *downloader* dapat digunakan perlu dilakukan instalasi *driver* terlebih dahulu. Konfigurasi *pinout* dan keterangan dari *downloader* terdapat pada Tabel 3.2 dan Gambar 3.3.

Tabel 3.2 Keterangan *pinout* AVR USB ISP

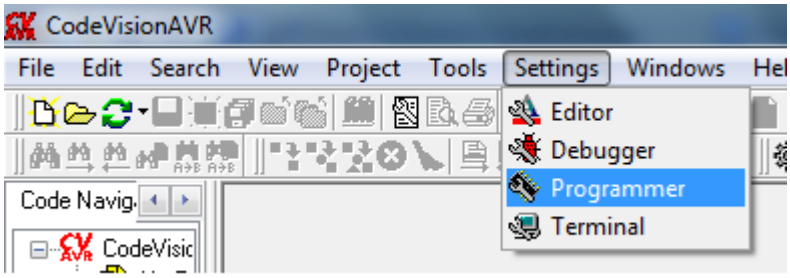
Nama	No. Pin	I/O	Keterangan
VTG	2	-	Catu daya dari target *board* (2.7 V - 5.5 V)
GND	4, 6, 8, 10	-	Titik referensi
LED	3	Output	Sinyal kontrol untuk LED (*Light Emitting Diode*) atau *multiplexer* (*optional*)
MOSI	1	Output	*Command* dan data dari AVR USB ISP ke target AVR
MISO	9	Input	Data dari target AVR ke AVR USB ISP
SCK	7	Output	Serial *Clock*, dikendalikan oleh AVR USB ISP
RESET	5	Output	*Reset*, dikendalikan oleh AVR USB ISP

Sumber: INNOVATIVE ELECTRONICS (2009)



Gambar 3.3 *Pinout* AVR USB ISP (INNOVATIVE ELECTRONICS, 2009)

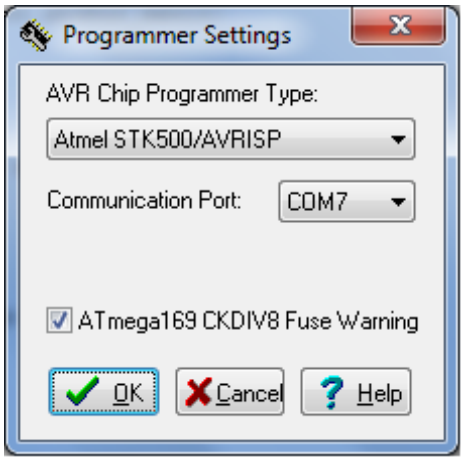
Pin MOSI, pin MISO, pin SCK, pin RESET, dan pin VTG pada AVR USB ISP masing-masing akan dihubungkan pada pin MOSI, pin MISO, pin SCK, pin RESET, dan pin VCC pada *microcontroller*. Program editor dan *compiler* yang digunakan untuk pembuatan program adalah *Code Vision* AVR. Proses *download* file ".hex" dapat dilakukan melalui program ini. Pengaturan penggunaan *downloader* pada *Code Vision* AVR dilakukan dengan memilih menu *Setting*, kemudian pilihan *Programmer* seperti yang ditunjukkan pada Gambar 3.4.



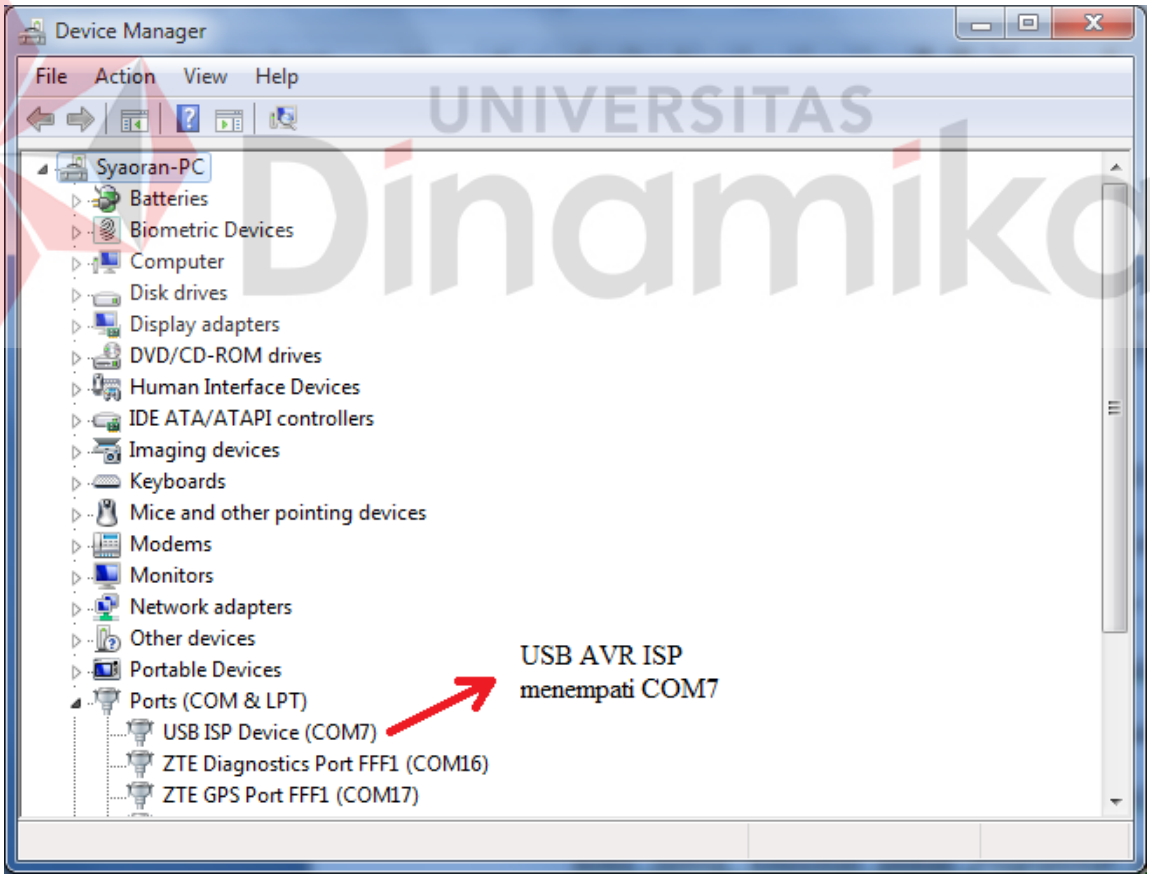
Gambar 3.4 Pemilihan *Programmer* pada menu *Setting* di *Code Vision* AVR

Setelah memilih *Programmer* pada menu *Setting*, akan muncul *window Programmer Setting* seperti pada Gambar 3.5, yang dilanjutkan dengan memilih tipe *programmer* AVR yaitu Atmel STK500/AVRISP. Pilihan *Communication Port* disesuaikan dengan nilai COM yang digunakan oleh *downloader*. Nilai COM

dari *downloader* dapat ditemukan pada *Device Manager* bagian *Ports* seperti pada Gambar 3.6.



Gambar 3.5 *Window Programmer Setting* pada *Code Vision* AVR



Gambar 3.6 *Device Manager*

B. Komunikasi Serial

Pengaturan mode operasi, *baud rate*, dan *frame format* harus dilakukan terlebih dahulu sebelum melakukan pengiriman ataupun penerimaan data secara serial. Mode operasi yang digunakan kali ini adalah pengiriman dan penerimaan data secara *asynchronous*, *baud rate* sebesar 9615 bps, dan 8-bit data *frame format*. Karena frekuensi osilator yang digunakan pada *microcontroller* adalah sebesar 4 MHz, maka *baud rate* yang dihasilkan tidak dapat persis bernilai 9600 bps. Perhitungannya adalah sebagai berikut:

$$UBRR = \frac{f_{osc}}{16 * BAUD} - 1 = \frac{4{,}000{,}000}{16 * 9{,}600} - 1 = \frac{4{,}000{,}000}{153{,}600} - 1 = 25.041667 \text{ .....} (3.1)$$

Karena pengaturan nilai UBRR tidak dapat dilakukan dengan angka desimal, maka nilai akhir pada persamaan 3.1 dibulatkan menjadi 25 atau 19h. Adapun *baud rate* yang didapatkan adalah:

$$BAUD = \frac{f_{osc}}{16 * (UBRR + 1)} = \frac{4{,}000{,}000}{16 * (25 + 1)} = \frac{4{,}000{,}000}{416} = 9{,}615.384615 bps \text{ ..... } (3.2)$$

Dengan hasil pada persamaan 3.2, maka nilai *error* yang dihasilkan adalah:

$$ERROR\% = \frac{9{,}615.384615 - 9600}{9600} * 100\% = \frac{15.384615}{9600} * 100\% = 0.16\% \text{ ..... } (3.3)$$

Karena sistem yang dibuat kali ini tidak membutuhkan pengiriman dan penerimaan data dalam ukuran yang besar, maka jumlah *error* sebesar 0.16% tidak akan memberikan pengaruh yang berarti, dengan kata lain dapat diabaikan. Rincian register terdapat pada Tabel 3.3, Tabel 3.4, Tabel 3.5, dan Tabel 3.6:

Tabel 3.3 Susunan bit dalam register UCSRA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
0	0	0	0	0	0	0	0

Tabel 3.4 Susunan bit dalam register UCSRB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
1	1	0	1	1	0	0	0

Tabel 3.5 Susunan bit dalam register UCSRC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
1	0	0	0	0	1	1	0

Tabel 3.6 Susunan bit dalam register UBRR

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
URSEL	-	-	-	UBRR[11:8]			
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	1
UBRR[7:0]							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Setelah nilai dari register-register konfigurasi komunikasi serial *microcontroller* didapatkan, maka dapat dilakukan inisialisasi pada program yaitu sebagai berikut:

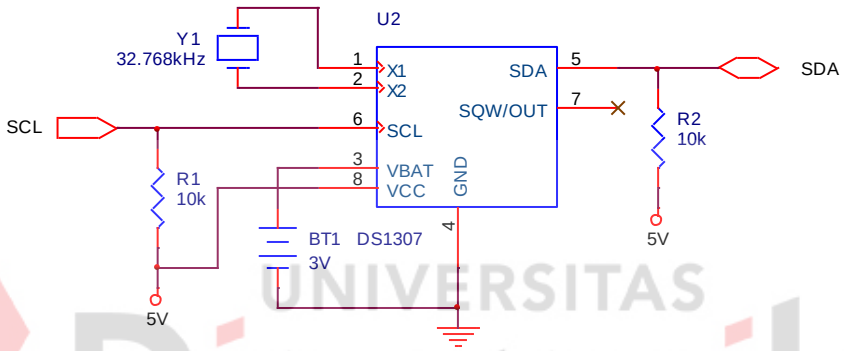
```
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;
```

Pengiriman dan penerimaan data serial dilakukan menggunakan fungsi sebagai berikut:

```
void putchar(char c) // pengiriman data char c (8-bit)
{
    while ((UCSRA & (1<<UDRE))==0); UDR=c;
}

char getchar(void) // penerimaan data 8-bit
{
    while ((UCSRA & (1<<RXC))==0); return UDR;
}
```

3.1.2. RTC DS1307

Rangkaian RTC DS1307 terdapat pada Gambar 3.7. Nilai tegangan dari catu daya utama yang digunakan berkisar antara 4.5 V - 5.5 V. Sedangkan catu daya cadangan (baterai) memiliki nilai tegangan 3 V. Jalur komunikasi I^2C yaitu, pin SDA dan pin SCL masing-masing diberi resistor *pull-up*. RTC membutuhkan rangkaian osilator eksternal yang hanya terdiri dari kristal dengan nilai frekuensi 32.768 kHz yang dihubungkan pada pin 1 dan pin 2 tanpa tambahan kapasitor.



Gambar 3.7 Skema rangkaian DS1307

Penggunaan RTC harus terlebih dahulu diawali dengan inisialisasi register *control*. Pengaturan register *control* RTC pada program dapat dilakukan dengan menggunakan fungsi sebagai berikut:

```
void rtc_init(unsigned char rs,unsigned char sqwe,unsigned char out)
{
    rs&=3;
    if (sqwe) rs|=0x10;
    if (out) rs|=0x80;
    i2c_start(); // kondisi start

    // 7-bit pertama untuk alamat DS1307 yaitu 0b1101000 yang diikuti dengan
    // 1-bit terakhir untuk arah data, 0 untuk penulisan dan 1 untuk pembacaan
    i2c_write(0xd0);

    i2c_write(7); // alamat register control DS1307
    i2c_write(rs); // menulis 8-bit data pada register control
    i2c_stop(); // kondisi stop
}
```

Pengaturan data waktu dan tanggal pada program dapat dilakukan dengan menggunakan fungsi-fungsi sebagai berikut:

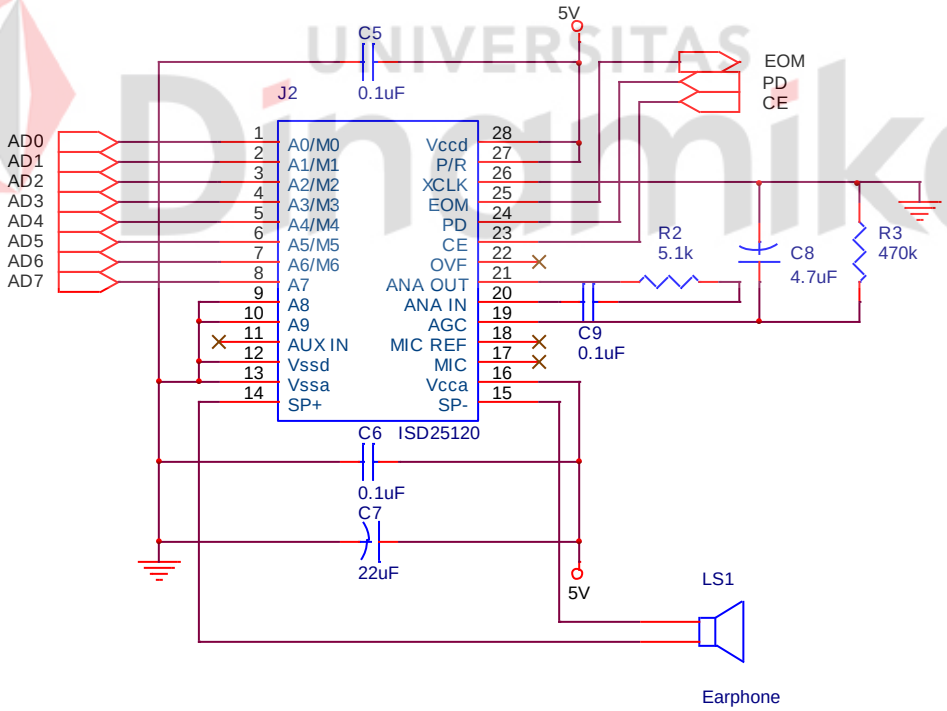
```
void rtc_set_time(unsigned char hour,unsigned char min,unsigned char sec)
{
    i2c_start(); // kondisi start
    i2c_write(0xd0); // melakukan operasi penulisan
    i2c_write(0); // menentukan alamat register yang akan ditulis yaitu 00h
    i2c_write(bin2bcd(sec)); // penulisan char sec pada register 00h
    i2c_write(bin2bcd(min)); // penulisan char min pada register 01h
    i2c_write(bin2bcd(hour)); // penulisan char hour pada register 02h
    i2c_stop(); // kondisi stop
}

void rtc_set_date(unsigned char date,unsigned char month,unsigned char year)
{
    i2c_start(); // kondisi start
    i2c_write(0xd0); // melakukan operasi penulisan
    i2c_write(4); // menentukan alamat register yang akan ditulis yaitu 04h
    i2c_write(bin2bcd(date)); // penulisan char date pada register 04h
    i2c_write(bin2bcd(month)); // penulisan char month pada register 05h
    i2c_write(bin2bcd(year)); // penulisan char year pada register 06h
    i2c_stop(); // kondisi stop
}
```

Pembacaan data waktu dan tanggal pada program dapat dilakukan dengan menggunakan fungsi-fungsi sebagai berikut:

```
void rtc_get_time(unsigned char *hour,unsigned char *min,unsigned char *sec)
{
    i2c_start(); // kondisi start
    i2c_write(0xd0); // melakukan operasi penulisan
    i2c_write(0); // menentukan alamat register yang akan dibaca yaitu 00h
    i2c_start(); // kondisi start
    i2c_write(0xd1); // melakukan operasi pembacaan
    *sec=bcd2bin(i2c_read(1)); // pembacaan data pada register 00h
    *min=bcd2bin(i2c_read(1)); // pembacaan data pada register 01h
    *hour=bcd2bin(i2c_read(0)); // pembacaan data pada register 02h
    i2c_stop(); // kondisi stop
}

void rtc_get_date(unsigned char *date,unsigned char *month,unsigned char *year)
{
    i2c_start(); // kondisi start
    i2c_write(0xd0); // menentukan operasi penulisan
    i2c_write(4); // menentukan alamat register yang akan dibaca yaitu 04h
    i2c_start(); // kondisi start
    i2c_write(0xd1); // melakukan operasi pembacaan
    *date=bcd2bin(i2c_read(1)); // pembacaan data pada register 04h
    *month=bcd2bin(i2c_read(1)); // pembacaan data pada register 05h
    *year=bcd2bin(i2c_read(0)); // pembacaan data pada register 06h
    i2c_stop();
}
```

3.1.3. ISD25120

Rangkaian ISD25120 terdapat pada Gambar 3.8. Pin 1 - pin 8 dihubungkan langsung dengan *port* A dari *microcontroller*. Nilai *port* A dari *microcontroller* digunakan sebagai masukan alamat operasi *playback* dan operasi *record* pada ISD25120. Pin 25 yaitu $\overline{EOM}$ akan menjadi masukan bagi *microcontroller* yang akan menandakan bahwa operasi *playback* pada alamat yang sesuai dengan nilai *port* A sudah selesai. Pin 24 yaitu PD akan diberi logika *high* saat ISD25120 tidak beroperasi agar konsumsi dayanya rendah. Operasi *playback* baru dapat berjalan apabila pin 23 yaitu $\overline{CE}$ telah diberi pulsa *low* pada saat pin PD berlogika *low*.



Gambar 3.8 Skema rangkaian ISD25120

Adapun pengoperasian rangkaian ISD25120 pada Gambar 3.8 dapat diringkas seperti pada Tabel 3.7.

Tabel 3.7 Langkah-langkah pengoperasian ISD25120

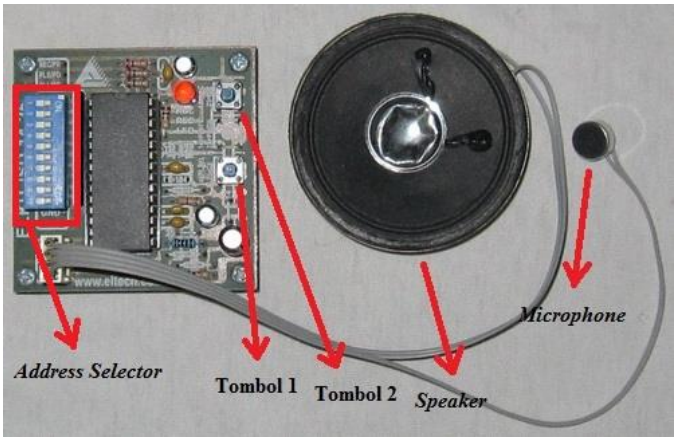
Control Step	*Function*	*Action*
1	*Power up chip and select Record/Playback mode*	1. PD = *low* 2. P/$\overline{R}$ = *as desired*
2	*Set message address for record/playback*	*Set addresses* 0 - 9
3A	*Begin Playback*	P/$\overline{R}$ = *high,* $\overline{CE}$ = *pulsed low*
3B	*Begin Record*	P/$\overline{R}$ = *low,* $\overline{CE}$ = *low*
4A	*End Playback*	*Automatic*
4B	*End Record*	PD *or* $\overline{CE}$ = *high*

Sumber: Information Storage Device (2000)

Penerapan langkah-langkah pengoperasian ISD25120 pada Tabel 3.7 dalam kode program adalah sebagai berikut:

```
void play(unsigned char isd_address)
{
    POWER_DOWN = 0 // step 1
    PORTA = isd_address; // step 2

    // step 3A
    CHIP_ENABLE = 0;
    delay_us(1);
    CHIP_ENABLE = 1;

    while(END_OF_MESSAGE == 1); // step 4A
    POWER_DOWN = 1; // mengembalikan ISD25120 pada konsumsi daya rendah
    delay_ms(25);
}
```

Proses *record* suara pada ISD25120 dilakukan dengan modul perekam yang terpisah. Modul perekam suara ini dapat dibeli di pasaran. Bentuk fisik dari modul ini terdapat pada Gambar 3.9. Proses *record* dilakukan dengan menekan dan menahan Tombol 2 kemudian dilanjutkan dengan menekan Tombol 1. Proses *record* akan berakhir ketika salah satu tombol dilepas. Proses *playback* dilakukan hanya dengan menekan Tombol 1.



Gambar 3.9 Alat perekam suara ISD

Terdapat 29 suara yang berbeda yang direkam pada ISD25120. Tabel 3.8 adalah ringkasan alamat akses dan keluaran suara ISD25120.

Tabel 3.8 Daftar alamat dan keluaran suara ISD25120

Alamat (desimal)	Keluaran Suara
0	Satu
4	Dua
8	Tiga
12	Empat
16	Lima
20	Enam
24	Tujuh
28	Delapan
32	Sembilan
36	Sepuluh
40	Sebelas
44	Pukul
48	Belas
52	Puluh
56	Menit
60	Lewat
64	Nol
68	Waktu
72	Sholat
76	Subuh
80	Dhuhur
84	Azhar
88	Maghrib
92	Isya’

Alamat (desimal)	Keluaran Suara
96	Kota
100	Pilihan
104	Surabaya
108	Jakarta
112	Denpasar

3.1.4. USB *to* Serial *Converter* CA-42

CA-42 digunakan untuk menghubungkan sistem yang dibuat dengan komputer. Gambar 3.10 menunjukkan *pinout* dari CA-42. Logika *high* pada CA-42 tidak direpresentasikan dengan tegangan 5 V melainkan dengan tegangan 3.3 V. Namun hal ini tidak akan mempengaruhi kinerjanya, karena suatu pin dianggap berlogika *high* oleh *microcontroller* pada saat tegangannya mencapai 0.6 * VCC. Karena tegangan VCC yang digunakan pada *microcontroller* adalah 5 V, maka saat pin *input* mendapat tegangan 3 V (0.6 * 5 V) maka pin tersebut dianggap berlogika high. Kabel yang terkoneksi dengan pin 6 yaitu RX akan dihubungkan dengan pin 15 pada *microcontroller*, sedangkan kabel yang terkoneksi dengan pin 7 yaitu TX akan dihubungkan dengan pin 14 pada *microcontroller*. Kabel yang terkoneksi dengan pin 8 yaitu GND akan dihubungkan dengan jalur *ground microcontroller*.

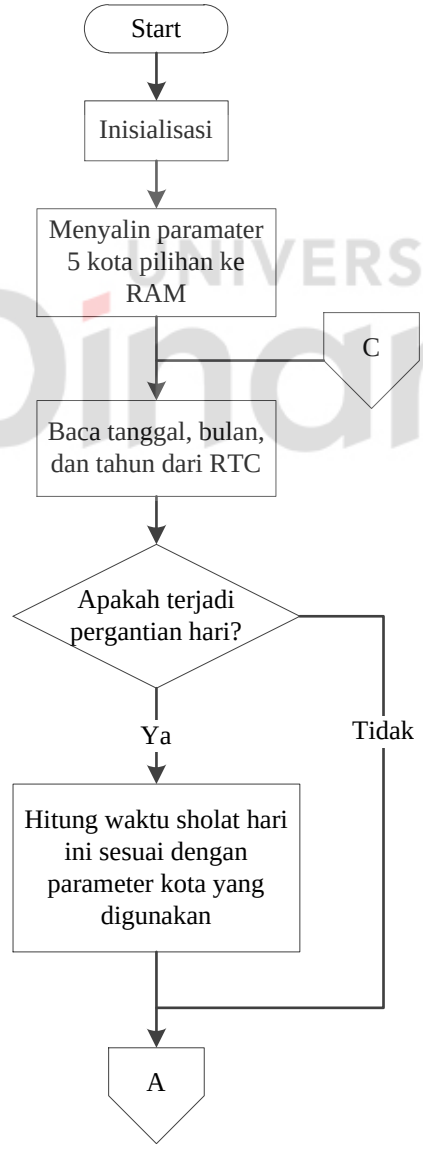
Gambar 3.10 *Pinout* CA-42 (Thomson, 2009)

3.2. Perancangan Perangkat Lunak

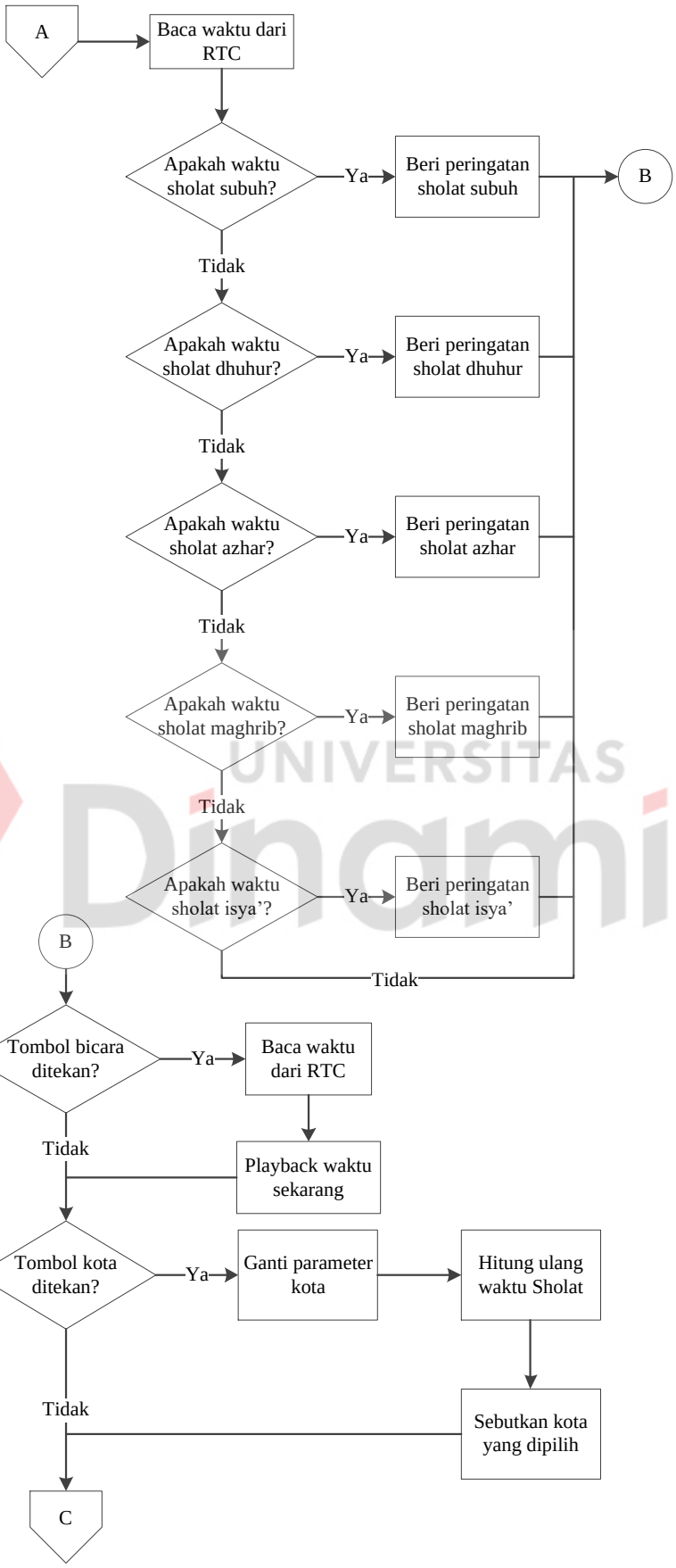
Perangkat lunak pada sistem ini dibagi menjadi dua yaitu perancangan perangkat lunak pada komputer dan perancangan perangkat lunak pada *microcontroller*.

3.2.1. Perangkat Lunak pada *Microcontroller*

Diagram alir perangkat lunak pada *microcontroller* terdapat pada Gambar 3.11 dan Gambar 3.12.



Gambar 3.11 Diagram alir program utama pada *microcontroller* bagian 1



Gambar 3.12 Diagram alir program utama pada *microcontroller* bagian 2

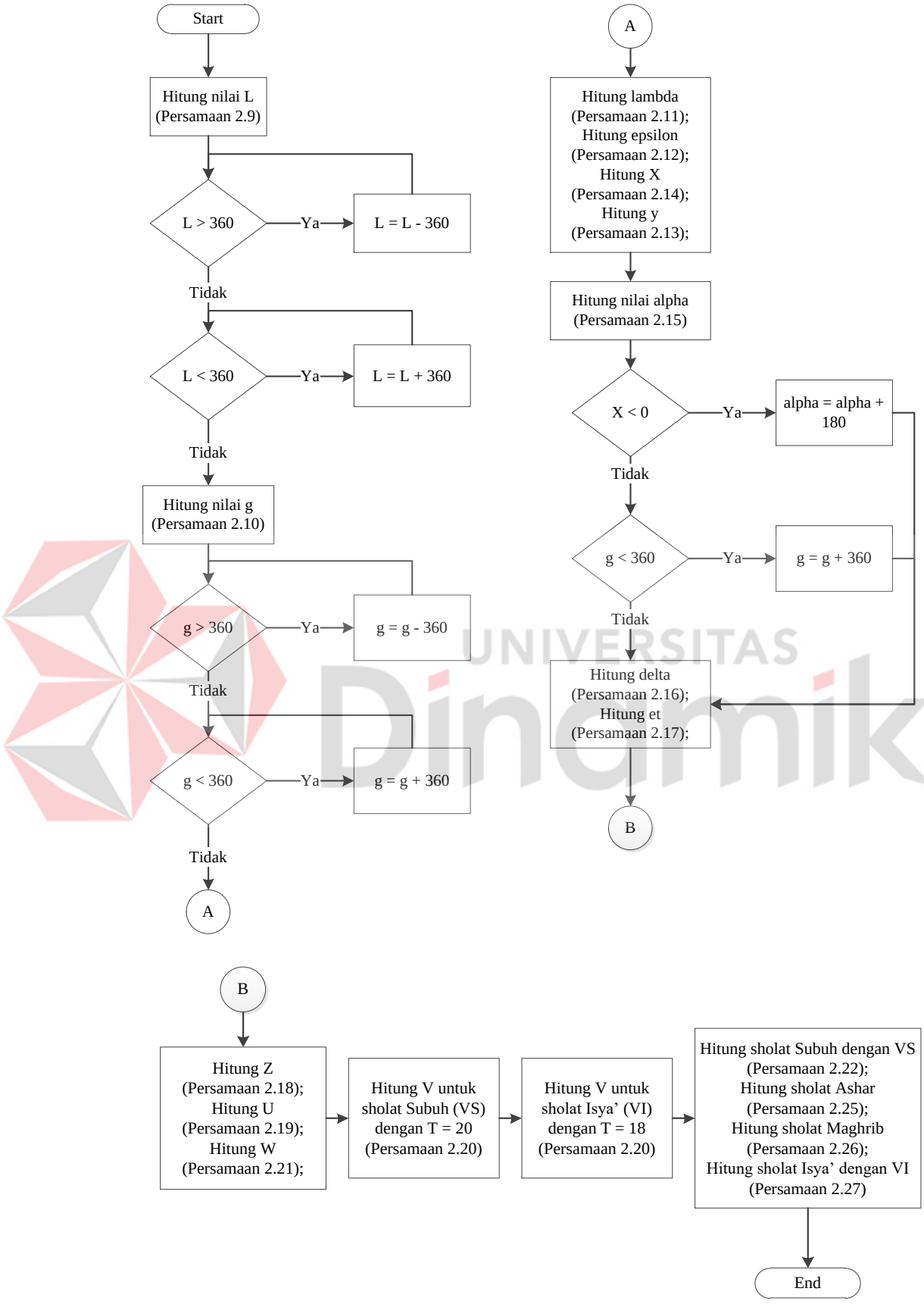
Instruksi pertama yang dijalankan oleh *microcontroller* adalah inisialisasi. Inisialisasi ini melingkupi register-register pada *microcontroller*, register-register pada RTC, dan variabel-variabel yang akan digunakan pada program. Instruksi berikutnya yang akan dieksekusi adalah melakukan penyalinan data dari EEPROM ke RAM. Hal ini dilakukan karena pembacaan data dari EEPROM relatif lebih lambat dibandingkan dengan pembacaan data dari RAM. Setelah penyalinan data dari EEPROM ke RAM, *microcontroller* akan melakukan pembacaan data tanggal, bulan, dan tahun dari RTC. Data-data tersebut digunakan sebagai pengambil keputusan apakah harus melakukan perhitungan waktu sholat atau tidak. Hal ini dilakukan agar perhitungan waktu sholat tidak dilakukan setiap saat, namun hanya sekali pada saat pergantian hari saja. Instruksi berikutnya yang akan dieksekusi oleh *microcontroller* adalah melakukan pembacaan data waktu dari RTC. Apabila data waktu tersebut sama dengan salah satu jadwal waktu sholat, maka *microcontroller* akan memberikan instruksi *playback* peringatan waktu sholat kepada ISD25120. Saat ISD25120 memberikan pulsa *low* pada pin EOM, maka *microcontroller* akan melakukan instruksi berikutnya yaitu mendeteksi penekanan tombol bicara. Apabila tombol bicara ditekan, maka *microcontroller* akan melakukan pembacaan data waktu pada RTC dan kemudian akan memberikan instruksi *playback* kepada ISD25120. Setelah *playback* oleh ISD25120 selesai, *microcontroller* akan mendeteksi penekanan tombol kota. Apabila tombol kota ditekan, *microcontroller* akan mengganti parameter kota yang digunakan, melakukan perhitungan ulang waktu sholat sesuai dengan parameter kota yang digunakan, dan kemudian memberikan instruksi *playback*

pada ISD25120. Program utama pada *microcontroller* yang disusun berdasarkan diagram alir pada Gambar 3.11 dan 3.12 adalah sebagai berikut:

```
void main()
{
    // variabel untuk pembacaan tanggal, bulan, dan tahun
    unsigned char date_t = 0, month_t = 0, year_t = 0;

    // variabel perulangan
    unsigned char looping;

    // inisialisasi port microcontroller
    PORTA=0x00; DDRA=0xFF;
    PORTB=0x06; DDRB=0xE1;
    PORTC=0xA0; DDRC=0x7F;
    PORTD=0xC0; DDRD=0x7C;

    // inisialisasi USART
    UCSRA=0x00; UCSRB=0xD8; UCSRC=0x86;
    UBRRH=0x00; UBRRL=0x19;

    i2c_init(); // inisialisasi I2C pada microcontroller
    rtc_init(0,1,0); // inisialisasi register control RTC

    #asm("sei") // mengaktifkan interupsi global

    // penyalinan parameter kota pilihan pada EEPROM ke RAM
    for(looping = 0; looping < 5; looping++)
    {
    H_kota[looping + BANYAK_KOTA_DEFINISI] = H_kota_pilihan[looping];
    bujur_kota[looping + BANYAK_KOTA_DEFINISI] = bujur_kota_pilihan[looping];
    lintang_kota[looping + BANYAK_KOTA_DEFINISI] = lintang_kota_pilihan[looping];
    gmt_kota[looping + BANYAK_KOTA_DEFINISI] = gmt_kota_pilihan[looping];
    }

    while(1)
    {
    // agar perhitungan waktu sholat tidak perlu dilakukan setiap saat
    rtc_get_date(&date_t, &month_t, &year_t);
    if(date_t!=tanggal||month_t!=bulan||year_t!=tahun)
    {
        tanggal = date_t; bulan = month_t; tahun = year_t;

        waktu_sholat(J2000(date_t, month_t, year_t), H_kota[index_kota],
        bujur_kota[index_kota], lintang_kota[index_kota], gmt_kota[index_kota]);

        flag_subuh = flag_dhuhur = flag_azhar = flag_maghrib = flag_isya = 0;
    }

    baca_waktu();

    if(!flag_subuh && h == subuh_jam && m == subuh_menit)
    {
        flag_subuh = 1;
        play(WAKTU);
        play(SHOLAT);
        play(SUBUH);
    }
    else if(!flag_dhuhur && h == dhuhur_jam && m == dhuhur_menit)
    {
        flag_dhuhur = 1;
        play(WAKTU);
        play(SHOLAT);
        play(DHUHUR);
    }
    else if(!flag_azhar && h == azhar_jam && m == azhar_menit)
    {
        flag_azhar = 1;
        play(WAKTU);
```

```
            play(SHOLAT);
            play(AZHAR);
        }
        else if(!flag_maghrib && h == maghrib_jam && m == maghrib_menit)
        {
            flag_maghrib = 1;
            play(WAKTU);
            play(SHOLAT);
            play(MAGHRIB);
        }
        else if(!flag_isya && h == isya_jam && m == isya_menit)
        {
            flag_isya = 1;
            play(WAKTU);
            play(SHOLAT);
            play(ISYA);
        }

        if(TALK_BUTTON == 0 || TALK_BUTTON2 == 0)
        {
            baca_waktu();
            play_puluhan_satuan(h, 0);
            if(m > 0)
                play_puluhan_satuan(m, 1);
            while(TALK_BUTTON == 0 || TALK_BUTTON2 == 0);
        }
        if(CITY_BUTTON == 0)
        {
            index_kota++;
            if(index_kota == BANYAK_KOTA_DEFINISI + 5)
                index_kota = 0;
            waktu_sholat(J2000(date_t, month_t, year_t), H_kota[index_kota],
            bujur_kota[index_kota], lintang_kota[index_kota], gmt_kota[index_kota]);

            play(KOTA);
            if(index_kota == 0)
                play(SURABAYA);
            else if(index_kota == 1)
                play(JAKARTA);
            else if(index_kota == 2)
                play(DENPASAR);
            else if(index_kota > 2)
            {
                play(PILIHAN);
                play(index_kota - 3);
            }
            while(CITY_BUTTON == 0);
        }
        }
}
```

Fungsi “waktu sholat(unsigned int k, float H, float bujur, float lintang, long int gmt)” adalah fungsi yang digunakan untuk melakukan perhitungan jadwal sholat lima waktu. Adapun diagram alir perhitungan waktu sholat terdapat pada Gambar 3.13.



Gambar 3.13 Diagram alir perhitungan waktu sholat

Kode program dari fungsi perhitungan waktu sholat yang dibuat berdasarkan persamaan 2.9 - persamaan 2.27 adalah sebagai berikut:

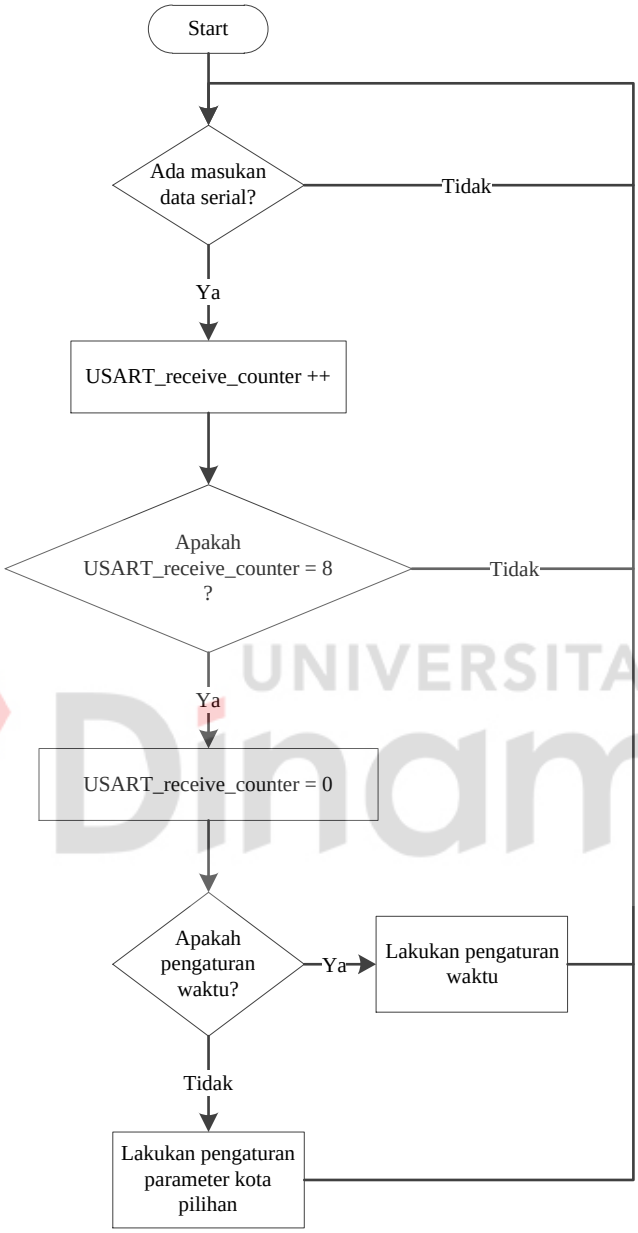
```
// k = J2000
// H = ketinggian kota
// bujur = bujur kota dalam bentuk desimal
// lintang = lintang kota dalam bentuk desimal
// gmt = zona waktu kota

void waktu_sholat(unsigned int k, float H, float bujur, float lintang, long int
gmt)
{
    float L, g;
    float lambda, epsilon;
    float X, Y;
    float alpha;
    float delta,et;
    float akar_ketinggian;
    float d, l;
    float cdl, sdl;
    float Z, U, VS, VI, c30, W;
    float subuh, dhuhur, azhar, maghrib, isya;

    // persamaan 2.9
    L = 280.461 + (0.9856474 * k);
    while(L > 360)
        L = L - 360;
    while(L < 0)
        L = L + 360;

    // persamaan 2.10
    g = 357.528 + (0.9856003 * k);
    while(g > 360)
        g = g - 360;
    while(g < 0)
        g = g + 360;

    // persamaan 2.11
    lambda = L+(1.915*sin(g*(PI/180)))+(0.02*sin((2*g)*(PI/180)));

    // persamaan 2.12
    epsilon = 23.439-(0.0000004*k);

    // persamaan 2.14
    X = cos(lambda * (PI / 180));

    // persamaan 2.13
    Y = cos(epsilon * (PI / 180)) * sin(lambda * (PI / 180));

    // persamaan 2.15
    alpha = atan(Y / X) * (180 / PI);
    if(X < 0)
        alpha = alpha + 180;
    else if(X > 0 && Y < 0)
        alpha = alpha + 360;

    // persamaan 2.16
    delta = asin(sin(epsilon*(PI/180))*sin(lambda*(PI/180)))*(180/PI);

    // persamaan 2.17
    et = (L - alpha) * 4;

    akar_ketinggian = sqrt(H);

    d = delta * (PI / 180);
    l = lintang * (PI / 180);

    cdl = cos(d) * cos(l);
    sdl = sin(d) * sin(l);
```

```
    // persamaan 2.18
    Z = 12+((gmt*15-bujur)/15)-(et/60);

    // persamaan 2.19
    U = (acos(((sin((-0.8333-0.0347*akar_ketinggian)*(PI/180)))-
         sdl)/cdl)*(180/PI))/15;

    // persamaan 2.20
    // Sudut twilight untuk subuh/fajar menurut TA mbak Ani 19, menurut PKPU 20
    VS = (acos((-sin(20*(PI/180))-sdl)/cdl)*(180/PI))/15; // untuk subuh
    VI = (acos((-sin(18*(PI/180))-sdl)/cdl)*(180/PI))/15; // untuk isya'


    // Angka 1 menurut syafi'i, 2 menurut hanafi
    c30 = 1+tan((fabs(lintang-delta))*(PI/180));

    c30 = sin((atan2(1,c30)*(180/PI))*(PI/180));

    // persamaan 2.21
    W = (acos((c30-sdl)/cdl)*(180/PI))/15;

    subuh = Z - VS; // persamaan 2.21
    dhuhur = Z; // persamaan 2.24
    azhar = Z + W; // persamaan 2.25
    maghrib = Z + U; // persamaan 2.26
    isya = Z + VI; // persamaan 2.27

    subuh_jam = subuh;
    dhuhur_jam = dhuhur;
    azhar_jam = azhar;
    maghrib_jam = maghrib;
    isya_jam = isya;

    subuh_menit = (subuh - subuh_jam) * 60;

    // Menurut saudari, Anik: dhuhur = ((dhuhur - dhuhur_jam)*60) + 3; ditambah 3
    // menit. Menurut PKPU tidak ditambah
    dhuhur_menit = ((dhuhur - dhuhur_jam) * 60);

    azhar_menit = (azhar - azhar_jam) * 60;
    maghrib_menit = (maghrib - maghrib_jam) * 60;
    isya_menit = (isya - isya_jam) * 60;
}
```

Penerimaan data serial dari komputer dilakukan dengan interupsi. Jika terdapat data pada register UDR yang belum dibaca, maka *microcontroller* akan meninggalkan rutin utama kemudian menjalankan rutin interupsi serial. Jika rutin interupsi serial sudah dilaksanakan maka *microcontroller* akan kembali melakukan eksekusi rutin utama. Adapun diagram alir dari rutin interupsi penerimaan data serial oleh *microcontroller* terdapat pada Gambar 3.14.

Pengaturan waktu dan pengaturan parameter kota pilihan membutuhkan 8 Byte data. Variabel "USART_receive_counter" digunakan untuk menghitung jumlah Byte data yang telah diterima. Jika data yang telah diterima telah

berjumlah 8 Byte maka pengaturan waktu dan pengaturan parameter kota pilihan dapat dilakukan.



Gambar 3.14 Diagram alir interupsi serial pada *microcontroller*

Adapun program interupsi serial pada *microcontroller* yang dibuat berdasarkan Gambar 3.14 adalah sebagai berikut:

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
    char status, data;
    status=UCSRA;
```

```
    data=UDR;
    if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
    {
        rx_buffer[rx_wr_index]=data;
        if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
        if (++rx_counter == RX_BUFFER_SIZE)
        {
            rx_counter=0;
            rx_buffer_overflow=1;
        };
    };

    /*** rutin interupsi ***/
    USART_receive_counter++;
    if(USART_receive_counter == 8)
    {
        // mengembalikan nilai USART_receive_counter menjadi 0
        // untuk proses penghitungan selanjutnya
        USART_receive_counter = 0;

        // pengaturan waktu
        if(rx_buffer[0] == 0) // Header data waktu
        {
            rtc_init(0,1,0);
            zona_waktu = rx_buffer[1];
            rtc_set_time(rx_buffer[2], rx_buffer[3], rx_buffer[4]);
            rtc_set_date(rx_buffer[5], rx_buffer[6], rx_buffer[7]);
        }

        // pengaturan parameter kota pilihan
        else
            hitung_parameter_kota_pilihan(rx_buffer[0] - 1);
    }
}
```

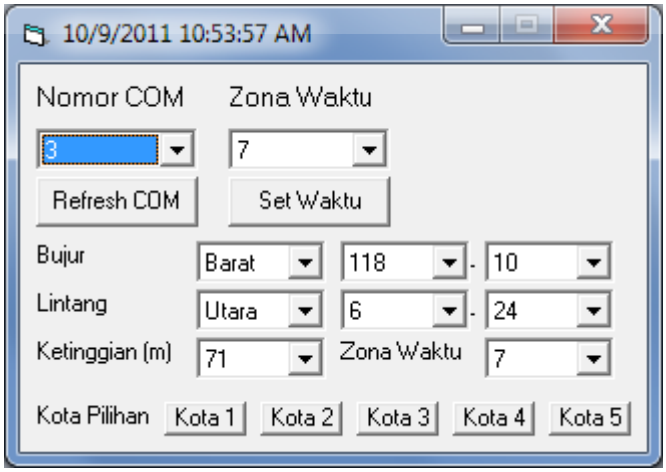
Fungsi “hitung_parameter_kota_pilihan(unsigned char i)” adalah fungsi yang digunakan untuk melakukan konversi bujur dan lintang dalam bentuk busur menjadi bentuk desimal. Kode program dari fungsi tersebut adalah sebagai berikut:

```
void hitung_parameter_kota_pilihan(unsigned char i)
{
    float tampung_bujur, tampung_lintang;
    unsigned char temp = 0;

    // karena rx_buffer mempunyai tipe data char (rentang nilai data -128 - 127)
    if(rx_buffer[2] > 127)
    {
        temp = rx_buffer[2] - 127;
        rx_buffer[2] = 127;
    }

    // sesuai dengan persamaan 2.1 - 2.4
    tampung_bujur = ((float)rx_buffer[2] + ((float)rx_buffer[3] / 60)) + temp;
    tampung_lintang = (float)rx_buffer[4] + (float)rx_buffer[5] / 60;
    if(rx_buffer[1] == 0) // barat & utara
    {
        bujur_kota_pilihan[i] = -1 * tampung_bujur;
        lintang_kota_pilihan[i] =  tampung_lintang;
    }
    else if(rx_buffer[1] == 1) // barat & selatan
    {
        bujur_kota_pilihan[i] = -1 * tampung_bujur;
```

```
        lintang_kota_pilihan[i] =  -1 * tampung_lintang;
    }
    else if(rx_buffer[1] == 2) // timur & utara
    {
        bujur_kota_pilihan[i] = tampung_bujur;
        lintang_kota_pilihan[i] =  tampung_lintang;
    }
    else // timur & selatan
    {
        bujur_kota_pilihan[i] = tampung_bujur;
        lintang_kota_pilihan[i] =  -1 * tampung_lintang;
    }
    H_kota_pilihan[i] = rx_buffer[6];
    gmt_kota_pilihan[i] = rx_buffer[7];

    // meng-update isi RAM
    H_kota[i + BANYAK_KOTA_DEFINISI] = H_kota_pilihan[i];
    bujur_kota[i + BANYAK_KOTA_DEFINISI] = bujur_kota_pilihan[i];
    lintang_kota[i + BANYAK_KOTA_DEFINISI] = lintang_kota_pilihan[i];
    gmt_kota[i + BANYAK_KOTA_DEFINISI] = gmt_kota_pilihan[i];
}
```

3.2.2. Perangkat Lunak pada Komputer

Komputer digunakan sebagai pengatur parameter yang dibutuhkan oleh sistem agar dapat bekerja dengan benar. Pengaturan yang diberikan meliputi pengaturan waktu (jam, menit, detik, tanggal, bulan, dan tahun) dan pengaturan parameter kota pilihan *user* (bujur, lintang, ketinggian, dan zona waktu). Tampilan program yang dibuat pada komputer ditunjukkan pada Gambar 3.15.



Gambar 3.15 Tampilan program pengatur parameter sistem pada komputer

BAB IV

PENGUJIAN SISTEM

Pengujian-pengujian yang dilakukan pada sistem adalah pengujian komunikasi serial antara sistem dengan komputer, pengujian pengaksesan RTC oleh *microcontroller*, pengujian pengaksesan dan keluaran suara dari ISD25120 melalui *earphone*, dan pengujian penjadwalan waktu sholat.

4.1. Pengujian Komunikasi Serial

4.1.1. Tujuan

Untuk mengetahui apakah komunikasi serial antara sistem dengan komputer dapat berlangsung dengan baik, yaitu pada saat pengiriman data maupun penerimaan data meskipun terdapat selisih *baud rate* antara sistem dengan komputer sebesar 15 bps.

4.1.2. Alat yang Digunakan

Peralatan yang dibutuhkan untuk pengujian ini adalah:

- Modul sistem
- Catu daya +9 V
- Seperangkat komputer dengan *port* USB
- USB *to* Serial *Converter* CA-42
- Program Terminal v1.9b.

4.1.3. Prosedur Pengujian

Langkah-langkah untuk melakukan pengujian komunikasi serial adalah sebagai berikut:

1. Berikan tegangan catu daya +9 V pada modul sistem, kemudian nyalakan sistem
2. *Download* program pengujian komunikasi serial berikut pada *microcontroller*

```
#include <mega32.h>
#include <stdio.h>

void main(void)
{
    // USART initialization
    // Communication Parameters: 8 Data, 1 Stop, No Parity
    // USART Receiver: On
    // USART Transmitter: On
    // USART Mode: Asynchronous
    // USART Baud Rate: 9600
    UCSRA=0x00;
    UCSRB=0x18;
    UCSRC=0x86;
    UBRRH=0x00;
    UBRRL=0x19;

    while (1)
        {
            // Place your code here
            putchar(getchar());
        };
}
```

3. Matikan catu daya
4. Hubungkan CA-42 dengan modul sistem dan komputer
5. Buka program Terminal lalu atur konfigurasinya sebagai berikut
   - *Port* : COM2 (sesuai dengan nilai pada *device manager*)
   - *Baud rate* : 9600
   - Data *bits* : 8
   - *Parity* : *none*
   - *Stop bits* : 1
   - Handhaking : *none*
6. Nyalakan catu daya pada modul sistem, kemudian nyalakan sistem
7. Kirimkan suatu karakter pada modul sistem melalui program Terminal

8. Dengan menggunakan program Terminal, lakukan pengamatan data serial yang diterima oleh komputer.

4.1.4. Hasil Pengujian

Pengujian komunikasi serial ini berjalan sesuai dengan harapan. Dari hasil pengujian, *error* pada *baud rate* sebesar 0.16% tidak mengakibatkan kesalahan pada pengiriman dan penerimaan data. Program yang di-*download* pada *microcontroller* bertujuan untuk menangkap data serial yang masuk, kemudian data tersebut dikirimkan kembali. Gambar 4.1 adalah hasil yang didapatkan dari pengamatan pada program Terminal.

Gambar 4.1 Hasil pengujian komunikasi serial

Ringkasan hasil pengiriman dan penerimaan data sebesar 8 Byte yang dilakukan sebanyak 30 kali menggunakan program Terminal terdapat pada Tabel 4.1.

Tabel 4.1 Ringkasan hasil pengujian komunikasi serial

Data yang dikirim	Data yang diterima
“SAtusaTU”	“SAtusaTU”
“ABCDabcd”	“ABCDabcd”
“TestTEST”	“TestTEST”
“efghEFGH”	“efghEFGH”
“pandaBAU”	“pandaBAU”
“bauPANDA”	“bauPANDA”
“ATmega32”	“ATmega32”
“HariRabu”	“HariRabu”
“qwerty-1”	“qwerty-1”
“QWERTY-2”	“QWERTY-2”
“pipiKiri”	“pipiKiri”
“25-10=15”	“25-10=15”
“25X4=100”	“25X4=100”
“PiZZaHUP”	“PiZZaHUP”
“McDomald”	“McDomald”
“sandalku”	“sandalku”
“SANDALMU”	“SANDALMU”
“merahAPi”	“merahAPi”
“bolabola”	“bolabola”
“hiJaUtua”	“hiJaUtua”
“birUmuda”	“birUmuda”
“lautmati”	“lautmati”
“ObatMata”	“ObatMata”
“gigiSusu”	“gigiSusu”
“esdingin”	“esdingin”
“cokaCOla”	“cokaCOla”
“sepatuku”	“sepatuku”
“SEPATUMU”	“SEPATUMU”
“BuKubUkU”	“BuKubUkU”
“manalime”	“manalime”

4.2. Pengujian RTC DS1307

4.2.1. Tujuan

Pengujian ini bertujuan untuk mengetahui apakah *microcontroller* dapat melakukan akses pada RTC DS1307 atau tidak, baik untuk mengubah data pada RTC maupun untuk membaca data dari RTC.

4.2.2. Alat yang Digunakan

Peralatan yang dibutuhkan untuk pengujian ini adalah:

- Modul sistem
- Catu daya +9 V
- Seperangkat komputer dengan *port* USB
- USB *to* Serial *Converter* CA-42
- Program Terminal v1.9b.

4.2.3. Prosedur Pengujian

Langkah-langkah untuk melakukan pengujian pada RTC DS1307 adalah sebagai berikut:

1. Berikan tegangan catu daya +9 V pada modul sistem, kemudian nyalakan sistem
2. *Download* program pengujian RTC berikut pada *microcontroller*

```
#include <mega32.h>

// I2C Bus functions
#asm
    .equ __i2c_port=0x18 ;PORTB
    .equ __sda_bit=4
    .equ __scl_bit=3
#endasm
#include <i2c.h>

#include <ds1307.h>
#include <stdio.h>
```

```
void main(void)
{
    unsigned char tanggal = 0, bulan = 0, tahun = 0;
    unsigned char tanggal_t = 0, bulan_t = 0, tahun_t = 0;
    unsigned char jam = 0, menit = 0, detik = 0;
    unsigned char jam_t = 0, menit_t = 0, detik_t = 0;

    // USART initialization
    // Communication Parameters: 8 Data, 1 Stop, No Parity
    // USART Receiver: On
    // USART Transmitter: On
    // USART Mode: Asynchronous
    // USART Baud Rate: 9600
    UCSRA=0x00;
    UCSRB=0x18;
    UCSRC=0x86;
    UBRRH=0x00;
    UBRRL=0x19;

    // I2C Bus initialization
    i2c_init();

    // DS1307 Real Time Clock initialization
    // Square wave output on pin SQW/OUT: Off
    // SQW/OUT pin state: 0
    rtc_init(0,0,0);

    // pengaturan tanggal: 9 Oktober 2011
    rtc_set_date(9, 10, 11);
    // pengaturan waktu: 21:00:00
    rtc_set_time(21, 0, 0);

    while (1)
        {
            rtc_get_date(&tanggal_t, &bulan_t, &tahun_t);
            if(tanggal_t != tanggal || bulan_t != bulan || tahun_t != tahun)
            {
                tanggal = tanggal_t;
                bulan = bulan_t;
                tahun = tahun_t;
                printf("Tanggal: %d-%d-%d\n", tanggal, bulan, tahun);
            }
            rtc_get_time(&jam_t, &menit_t, &detik_t);
            if(jam_t != jam || menit_t != menit || detik_t != detik)
            {
                jam = jam_t;
                menit = menit_t;
                detik = detik_t;
                printf("Waktu: %d:%d:%d\n", jam, menit, detik);
            }
        };
}
```

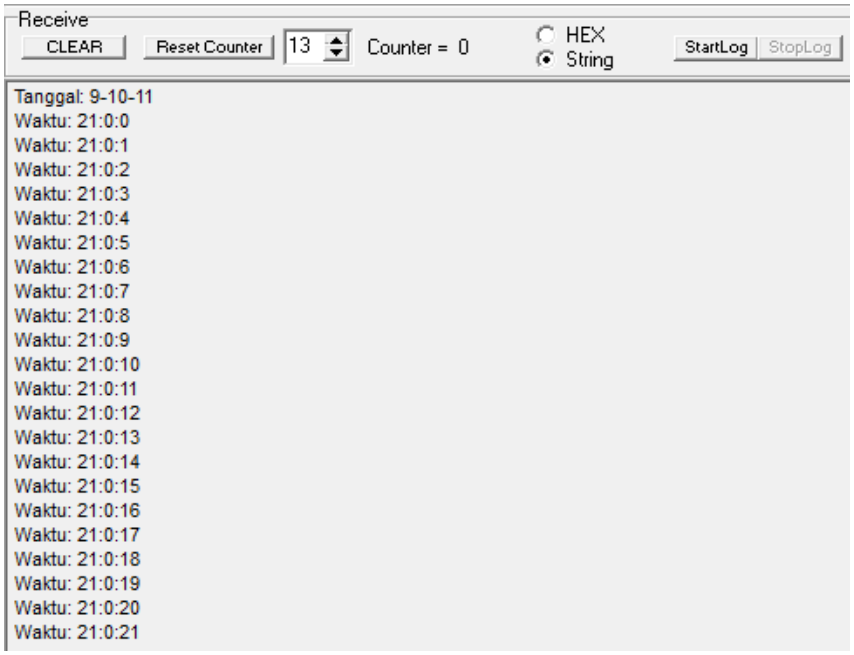
3. Matikan catu daya
4. Hubungkan CA-42 dengan modul sistem dan komputer
5. Buka program Terminal lalu atur konfigurasinya sebagai berikut
   - *Port* : COM2 (sesuai dengan nilai pada *device manager*)
   - *Baud rate* : 9600
   - Data *bits* : 8

- *Parity* : *none*
- *Stop bits* : 1
- Handhaking : *none*

6. Nyalakan catu daya pada modul sistem, kemudian nyalakan sistem
7. Dengan menggunakan program Terminal, lakukan pengamatan data serial yang diterima oleh komputer.

4.2.4. Hasil Pengujian

Pengujian RTC DS1307 ini berjalan sesuai dengan harapan. Program yang di-*download* pada *microcontroller* pertama-tama akan melakukan inisialisasi register *control* pada RTC yang kemudian akan dilanjutkan dengan inisialisasi waktu yaitu tanggal, bulan, tahun, jam, menit, dan detik. Setelah proses inisialisasi selesai, *microcontroller* akan melakukan pembacaan data pada RTC secara terus menerus yang kemudian akan dikirimkan pada komputer secara serial apabila terdapat perubahan data.



Gambar 4.2. Data waktu dari RTC yang dibaca oleh *microcontroller*

Gambar 4.2 adalah tampilan dari program Terminal pada saat pengujian dilakukan. Data waktu pertama diterima pukul 01:52:44 waktu yang sebenarnya dan waktu yang dibaca oleh *microcontroller* dari RTC adalah pukul 21:00:00, 1 detik kemudian, yaitu pukul 01:52:45 waktu yang sebenarnya, waktu yang dibaca oleh *microcontroller* berubah menjadi pukul 21:00:01, dan seterusnya. Dapat disimpulkan bahwa perubahan waktu di RTC sama dengan perubahan waktu sebenarnya.

4.3. Pengujian ISD25120

4.3.1. Tujuan

Pengujian ini bertujuan untuk mengetahui apakah *microcontroller* dapat mengakses ISD25120 atau tidak, apakah suara yang direkam memiliki urutan yang sama dengan Tabel 3.8, serta melakukan pengujian apakah keluaran suara dari ISD25120 pada *earphone* terdengar jelas oleh *user* atau tidak.

4.3.2. Alat yang Digunakan

Peralatan yang dibutuhkan untuk pengujian ini adalah:

- Modul sistem
- Catu daya +9 V
- *Earphone*
- Seperangkat komputer dengan *port* USB.

4.3.3. Prosedur Pengujian

Langkah-langkah untuk melakukan pengujian pada ISD25120 adalah sebagai berikut:

1. Berikan tegangan catu daya +9 V pada modul sistem, kemudian nyalakan sistem
2. *Download* program pengujian ISD25120 berikut pada *microcontroller*

```
#include <mega32.h>
#include <delay.h>

#define END_OF_MESSAGE PINC.7
#define POWER_DOWN PORTC.6
#define CHIP_ENABLE PORTC.5

/*****************************************************
ISD Address Map

Address         Output          Address         Output
0               '1'             20              '6'
4               '2'             24              '7'
8               '3'             28              '8'
12              '4'             32              '9'
16              '5'             36              '10'

Address         Output          Address         Output
40              '11'            60              "lewat"
44              "pukul"         64              '0'
48              "belas"         68              "waktu"
52              "puluh"         72              "sholat"
56              "menit"         76              "subuh"

Address         Output          Address         Output
80              "dhuhur"        100             "pilihan"
84              "azhar"         104             "Surabaya"
88              "maghrib"       108             "Jakarta"
92              "isya'"         112             "Denpasar"
96              "kota"
*****************************************************/

void play(unsigned char index)
{
    POWER_DOWN = 0;
    PORTA = index * 4;

    CHIP_ENABLE = 0;
    delay_us(1);
    CHIP_ENABLE = 1;

    while(END_OF_MESSAGE == 1);
    POWER_DOWN = 1;
    delay_ms(50);
}

void main(void)
{
    unsigned char looping = 0;

    // Port A initialization
    PORTA=0x00;
    DDRA=0xFF;
    // Port C initialization
    PORTC=0xA0;
    DDRC=0x7F;

    while (1)
        {
            for(looping = 0; looping < 29; looping++)
            {
                play(looping); delay_ms(500);
            }
        };
}
```

3. Matikan catu daya
4. Hubungkan *earphone* dengan modul sistem
5. Nyalakan catu daya pada modul sistem, kemudian nyalakan sistem
6. Dengarkan dan amati urutan keluaran suara pada *earphone*, apakah suara yang dikeluarkan jelas dan sesuai dengan urutan perekaman suara seperti pada Tabel 3.8.

4.3.4. Hasil Pengujian

Pengujian ISD25120 ini berjalan sesuai dengan harapan. Program yang di-*download* pada *microcontroller* akan memberikan instruksi *playback* kepada ISD25120 mulai dari rekaman suara pada alamat 0 sampai rekaman suara pada alamat 112. Urutan keluaran suara oleh ISD25120 pada *earphone* sesuai dengan Tabel 3.8 dan terdengar jelas. Untuk menghindari penilaian mengenai kejelasan keluaran suara secara subjektif, maka penulis melakukan survei kepada sepuluh orang. Ringkasan dari survei tersebut terdapat pada Tabel 4.2

Tabel 4.2 Ringkasan survei mengenai kejelasan keluaran suara ISD25120

Responden ke-	Jelas/Tidak	Keterangan
1	Jelas	Terdapat *background* suara mendengung
2	Jelas	-
3	Jelas	Terdapat *background* suara mendengung
4	Jelas	Terdapat *background* suara mendengung
5	Jelas	Terdapat *background* suara mendengung
6	Jelas	Terdapat *background* suara mendengung
7	Jelas	-
8	Jelas	Terdapat *background* suara mendengung
9	Jelas	-
10	Jelas	Terdapat *background* suara mendengung

4.4. Pengujian Penjadwalan Waktu Sholat

4.4.1. Tujuan

Pengujian ini bertujuan untuk mengetahui apakah *microcontroller* dapat melakukan penjadwalan waktu sholat dengan tepat atau tidak.

4.4.2. Alat yang Digunakan

- Modul sistem
- Catu daya +9 V
- Seperangkat komputer dengan *port* USB
- USB *to* Serial *Converter* CA-42
- Program pengatur waktu dan parameter kota
- Program Terminal v1.9b

4.4.3. Prosedur Pengujian

Langkah-langkah untuk melakukan pengujian penjadwalan waktu sholat adalah sebagai berikut:

1. Berikan tegangan catu daya +9 V pada modul sistem, kemudian nyalakan sistem
2. *Download* program pengujian penjadwalan waktu sholat berikut pada *microcontroller*

```
#include <mega32.h>
#include <delay.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>

#define BANYAK_KOTA_DEFINISI 3
#define TALK_BUTTON PINB.1
#define TALK_BUTTON2 PIND.7
#define CITY_BUTTON PINB.2

// variabel counter untuk menghitung penerimaan data
unsigned char USART_receive_counter;
// index penanda kota pilihan pengguna saat ini
```

```
eeprom unsigned char index_kota = 0;


float H_kota[BANYAK_KOTA_DEFINISI + 5] = {5, 1, 40, 0, 0, 0, 0, 0};

float   bujur_kota[BANYAK_KOTA_DEFINISI   +   5]   =   {112.7425,   106.845172,
115.222099, 0, 0, 0, 0, 0};

float  lintang_kota[BANYAK_KOTA_DEFINISI  +  5]  =  {-7.265278,  -6.211544,  -
8.65629, 0, 0, 0, 0, 0};

long int gmt_kota[BANYAK_KOTA_DEFINISI + 5] = {7, 7, 8, 0, 0, 0, 0, 0};

/*Parameter kota pilihan*/
eeprom float H_kota_pilihan[5] = {0, 0, 0, 0, 0};
eeprom float bujur_kota_pilihan[5] = {0, 0, 0, 0, 0};
eeprom float lintang_kota_pilihan[5] = {0, 0, 0, 0, 0};
eeprom long int gmt_kota_pilihan[5] = {0, 0, 0, 0, 0};

unsigned char subuh_jam, dhuhur_jam, azhar_jam, maghrib_jam, isya_jam;
unsigned   char   subuh_menit,   dhuhur_menit,   azhar_menit,   maghrib_menit,
isya_menit;

unsigned  int  J2000(unsigned  char  date,  unsigned  char  month,  unsigned  char
year)
{
    unsigned int i, j2000 = 0;
    for(i = (year+2000)-1; i >= 2000; i--)
    {
        if(i%400 == 0)
            j2000 = j2000 + 366;
        else if(i%100 == 0)
            j2000 = j2000 + 365;
        else if(i%4 == 0)
            j2000 = j2000 + 366;
        else
            j2000 = j2000 + 365;
    }
    for(i = month-1; i >= 1; i--)
    {
        if(i == 2)
        {
            if((year+2000)%400 == 0)
                j2000 = j2000 + 29;
            else if((year+2000)%100 == 0)
                j2000 = j2000 + 28;
            else if((year+2000)%4 == 0)
                j2000 = j2000 + 29;
            else
                j2000 = j2000 + 28;
        }
        else if(i==1||i== 3||i == 5||i == 7||i == 8||i == 10)
            j2000 = j2000 + 31;
        else
            j2000 = j2000 + 30;
    }
    j2000 = j2000 + date - 1;
    return j2000;
}

void  waktu_sholat(unsigned  int  k,  float  H,  float  bujur,  float  lintang,  long
int gmt)
{
    float L, g;
    float lambda, epsilon;
    float X, Y;
    float alpha;
    float delta,et;
    float akar_ketinggian;
    float d, l;
    float cdl, sdl;
    float Z, U, VS, VI, c30, W;
    float subuh, dhuhur, azhar, maghrib, isya;
```

```
    L = 280.461 + (0.9856474 * k);
    while(L > 360)
        L = L - 360;
    while(L < 0)
        L = L + 360;

    g = 357.528 + (0.9856003 * k);
    while(g > 360)
        g = g - 360;
    while(g < 0)
        g = g + 360;

    lambda = L+(1.915*sin(g*(PI/180)))+(0.02*sin((2*g)*(PI/180)));
    epsilon = 23.439-(0.0000004*k);

    X = cos(lambda * (PI / 180));
    Y = cos(epsilon * (PI / 180)) * sin(lambda * (PI / 180));

    alpha = atan(Y / X) * (180 / PI);
    if(X < 0)
        alpha = alpha + 180;
    else if(X > 0 && Y < 0)
        alpha = alpha + 360;

    delta = asin(sin(epsilon*(PI/180))*sin(lambda*(PI/180)))*(180/PI);
    et = (L - alpha) * 4;

    akar_ketinggian = sqrt(H);

    d = delta * (PI / 180);
    l = lintang * (PI / 180);

    cdl = cos(d) * cos(l);
    sdl = sin(d) * sin(l);

    Z = 12+((gmt*15-bujur)/15)-(et/60);
    U = (acos(((sin((-0.8333-0.0347*akar_ketinggian)*(PI/180)))-
    sdl)/cdl)*(180/PI))/15;
    VS = (acos((-sin(20*(PI/180))-sdl)/cdl)*(180/PI))/15;
    VI = (acos((-sin(18*(PI/180))-sdl)/cdl)*(180/PI))/15;
    c30 = 1+tan((fabs(lintang-delta))*(PI/180));
    c30 = sin((atan2(1,c30)*(180/PI))*(PI/180));
    W = (acos((c30-sdl)/cdl)*(180/PI))/15;

    subuh = Z - VS;
    dhuhur = Z;
    azhar = Z + W;
    maghrib = Z + U;
    isya = Z + VI;

    subuh_jam = subuh;
    dhuhur_jam = dhuhur;
    azhar_jam = azhar;
    maghrib_jam = maghrib;
    isya_jam = isya;

    subuh_menit = (subuh - subuh_jam) * 60;
    dhuhur_menit = ((dhuhur - dhuhur_jam) * 60);
    azhar_menit = (azhar - azhar_jam) * 60;
    maghrib_menit = (maghrib - maghrib_jam) * 60;
    isya_menit = (isya - isya_jam) * 60;
}

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
```

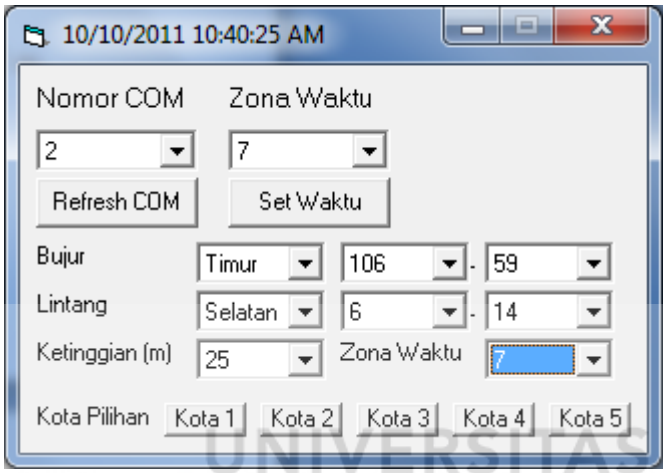
```
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

void hitung_parameter_kota_pilihan(unsigned char i)
{
    float tampung_bujur, tampung_lintang;
    unsigned char temp = 0;
    if(rx_buffer[2] > 127) // karena rx_buffer mempunyai tipe data char
    {
        temp = rx_buffer[2] - 127;
        rx_buffer[2] = 127;
    }

    tampung_bujur = ((float)rx_buffer[2] + ((float)rx_buffer[3] / 60)) + temp;
    tampung_lintang = (float)rx_buffer[4] + (float)rx_buffer[5] / 60;
    if(rx_buffer[1] == 0) // barat & utara
    {
        bujur_kota_pilihan[i] = -1 * tampung_bujur;
        lintang_kota_pilihan[i] =  tampung_lintang;
    }
    else if(rx_buffer[1] == 1) // barat & selatan
    {
        bujur_kota_pilihan[i] = -1 * tampung_bujur;
        lintang_kota_pilihan[i] =  -1 * tampung_lintang;
    }
    else if(rx_buffer[1] == 2) // timur & utara
    {
        bujur_kota_pilihan[i] = tampung_bujur;
        lintang_kota_pilihan[i] =  tampung_lintang;
    }
    else // timur & selatan
    {
        bujur_kota_pilihan[i] = tampung_bujur;
        lintang_kota_pilihan[i] =  -1 * tampung_lintang;
    }
    H_kota_pilihan[i] = rx_buffer[6];
    gmt_kota_pilihan[i] = rx_buffer[7];

    // meng-update isi RAM
    H_kota[i + BANYAK_KOTA_DEFINISI] = H_kota_pilihan[i];
    bujur_kota[i + BANYAK_KOTA_DEFINISI] = bujur_kota_pilihan[i];
    lintang_kota[i + BANYAK_KOTA_DEFINISI] = lintang_kota_pilihan[i];
    gmt_kota[i + BANYAK_KOTA_DEFINISI] = gmt_kota_pilihan[i];
}

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
    char status, data;
    status=UCSRA;
    data=UDR;
    if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
        {
            rx_buffer[rx_wr_index]=data;
            if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
            if (++rx_counter == RX_BUFFER_SIZE)
            {
                rx_counter=0;
                rx_buffer_overflow=1;
            };
        };
```

```
    // rutin interupsi
    USART_receive_counter++;
    if(USART_receive_counter == 8)
    {
        // mengembalikan nilai USART_receive_counter menjadi 0 untuk proses
        //penghitungan selanjutnya
        USART_receive_counter = 0;

        // pengaturan parameter kota pilihan
        hitung_parameter_kota_pilihan(rx_buffer[0] - 1);
    }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

void main(void)
{
    unsigned char looping = 0;

    // Port B initialization
    PORTB=0x06;
    DDRB=0x00;
    // Port D initialization
    PORTD=0x80;
    DDRD=0x00;

    // USART initialization
    // Communication Parameters: 8 Data, 1 Stop, No Parity
    // USART Receiver: On
    // USART Transmitter: On
    // USART Mode: Asynchronous
    // USART Baud Rate: 9600
    UCSRA=0x00;
    UCSRB=0x98;
    UCSRC=0x86;
    UBRRH=0x00;
    UBRRL=0x19;


    // Global enable interrupts
    #asm("sei")

    // penyalinan parameter kota pilihan pada EEPROM ke RAM
    for(looping = 0; looping < 5; looping++)
    {
        H_kota[looping + BANYAK_KOTA_DEFINISI] = H_kota_pilihan[looping];

        bujur_kota[looping + BANYAK_KOTA_DEFINISI] =
        bujur_kota_pilihan[looping];

        lintang_kota[looping + BANYAK_KOTA_DEFINISI] =
        lintang_kota_pilihan[looping];

        gmt_kota[looping + BANYAK_KOTA_DEFINISI] = gmt_kota_pilihan[looping];
    }

    while (1)
```

```
{
    if(TALK_BUTTON == 0 || TALK_BUTTON2 == 0)
    {
    for(looping = 1; looping < 32; looping++)
    {
    waktu_sholat(J2000(looping, 10, 11), H_kota[index_kota],
    bujur_kota[index_kota], lintang_kota[index_kota],
    gmt_kota[index_kota]);

    printf("Tanggal: %d\n", looping);
    printf("Subuh: %d:%d ", subuh_jam, subuh_menit);
    printf("Dhuhur: %d:%d ", dhuhur_jam, dhuhur_menit);
    printf("Azhar: %d:%d ", azhar_jam, azhar_menit);
    printf("Maghrib: %d:%d ", maghrib_jam, maghrib_menit);
    printf("Isya': %d:%d\n", isya_jam, isya_menit);
    }

    while(TALK_BUTTON == 0 || TALK_BUTTON2 == 0);
    }
    if(CITY_BUTTON == 0)
    {
    index_kota++;
    if(index_kota == BANYAK_KOTA_DEFINISI + 5)
        index_kota = 0;
    printf("Index: %d\n", index_kota);

    while(CITY_BUTTON == 0);
    }
};
}
```

3. Matikan catu daya
4. Hubungkan CA-42 dengan modul sistem dan komputer
5. Nyalakan catu daya pada modul sistem, kemudian nyalakan sistem
6. Buka program pengatur waktu dan parameter kota lalu lakukan langkah-langkah berikut:
    a. Atur nilai pada kolom “Nomor COM” menjadi 2 (sesuai dengan nilai pada *device manager*)
    b. Atur nilai-nilai kolom “Bujur”, “Lintang”, “Ketinggian”, dan “Zona Waktu” sesuai dengan yang diinginkan
    c. Tekan tombol “Kota 1” untuk mengirimkan parameter kota yang sudah diatur pada langkah b sebagai parameter kota ke-1, tombol “Kota 2” untuk parameter kota ke-2, dst.
    d. Tutup program pengatur waktu dan parameter kota

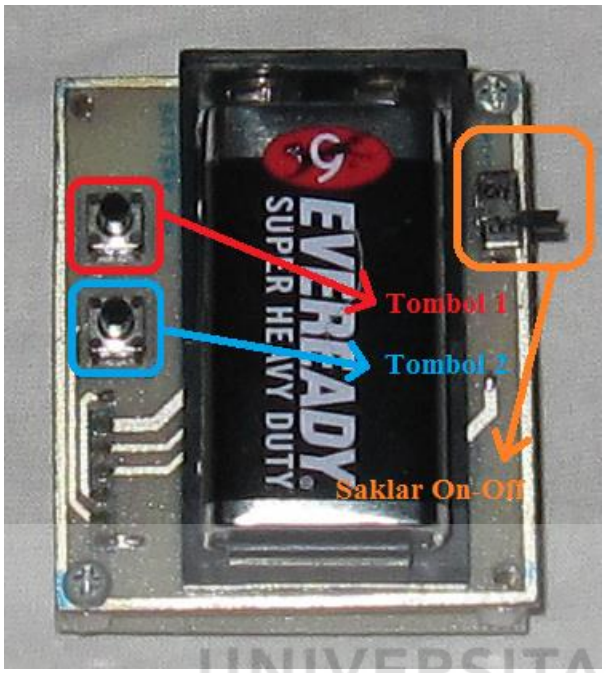
Pada pengujian ini, kota yang parameternya dimasukkan ke dalam *microcontroller* adalah kota Bekasi (106°59 BT, 6°14 LS, Ketinggian: 25 m, Zona Waktu: 7) sebagai kota ke-1 dan kota Pontianak (109°19 BT, 0°0 LS, Ketinggian: 1 m, Zona Waktu: 7) sebagai kota ke-2. Adapun tampilan dari program pengatur waktu dan parameter kota terdapat pada Gambar 4.3



Gambar 4.3 Tampilan program pengatur waktu dan parameter kota

7. Buka program Terminal lalu atur konfigurasinya sebagai berikut:
   - *Port* : COM2 (sesuai dengan nilai pada *device manager*)
   - *Baud rate* : 9600
   - Data *bits* : 8
   - *Parity* : *none*
   - *Stop bits* : 1
   - Handhaking : *none*
8. Tekan Tombol 1 untuk melakukan pergantian kota, Tekan Tombol 2 untuk melakukan perhitungan waktu sholat. Adapun peletakan Tombol 1 dan Tombol 2 pada modul sistem terdapat pada Gambar 4.4

9. Dengan menggunakan program Terminal, lakukan pengamatan data serial yang diterima oleh komputer.

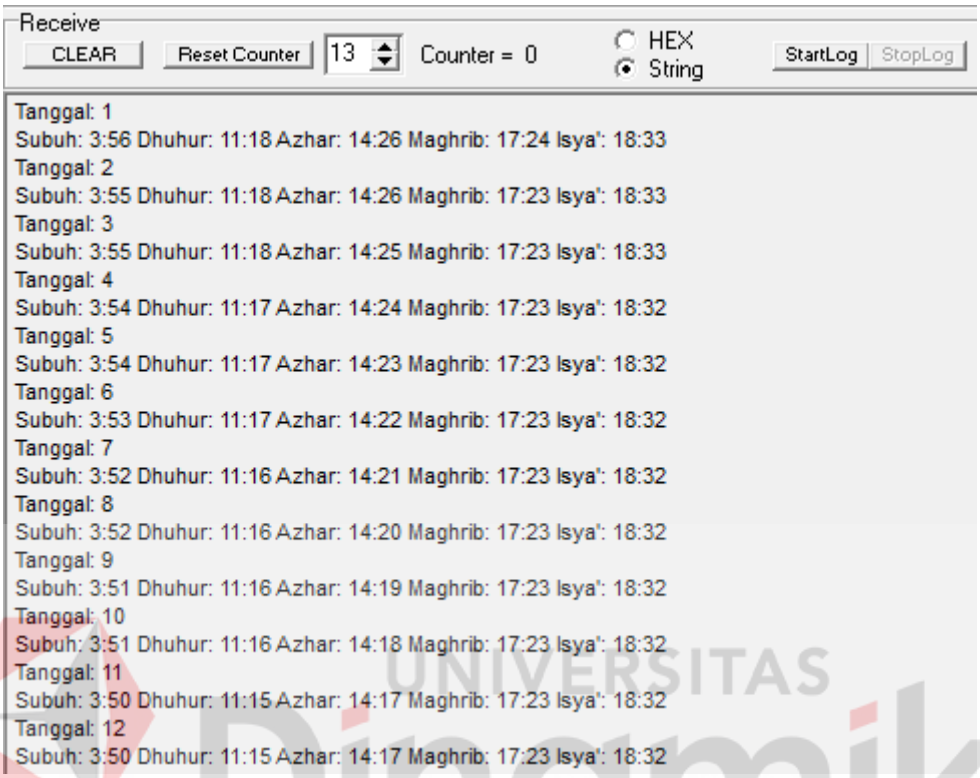


Gambar 4.4 Modul sistem

4.4.4. Hasil Pengujian

Dari hasil pengujian penjadwalan waktu sholat ini didapatkan selisih waktu antara 1 sampai 3 menit untuk beberapa waktu sholat. Hasil dari penjadwalan sholat oleh *microcontroller* dibandingkan dengan jadwal waktu sholat yang didapatkan dari situs PKPU yaitu http://www.pkpu.or.id/adzan.lite.php. Program yang di-*download* pada *microcontroller* akan melakukan penjadwalan waktu sholat untuk lima kota yaitu Surabaya, Jakarta, Denpasar, Bekasi, dan Pontianak mulai dari tanggal 1 sampai tanggal 31 pada bulan Oktober tahun 2011. Hasil dari penjadwalan waktu sholat tersebut akan dikirimkan secara serial pada komputer. Adapun tampilan dari

program Terminal saat dilakukan pengujian terdapat pada Gambar 4.5. Ringkasan dari pengujian yang dilakukan terdapat pada Tabel 4.3 sampai dengan Tabel 4.7.



Gambar 4.5 Tampilan program Terminal pada saat pengujian penjadwalan waktu sholat

Tabel 4.3 Perbandingan jadwal waktu sholat kota Surabaya bulan Oktober 2011

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
1	03:56	11:18	14:26	17:24	18:33	03:56	11:19	14:27	17:23	18:33
2	03:55	11:18	14:26	17:23	18:33	03:56	11:18	14:26	17:23	18:33
3	03:55	11:18	14:25	17:23	18:33	03:55	11:18	14:26	17:23	18:33
4	03:54	11:17	14:24	17:23	18:32	03:55	11:18	14:25	17:23	18:32
5	03:54	11:17	14:23	17:23	18:32	03:54	11:17	14:24	17:23	18:32
6	03:53	11:17	14:22	17:23	18:32	03:53	11:17	14:23	17:23	18:32
7	03:52	11:16	14:21	17:23	18:32	03:53	11:17	14:22	17:23	18:32
8	03:52	11:16	14:20	17:23	18:32	03:52	11:17	14:21	17:23	18:32
9	03:51	11:16	14:19	17:23	18:32	03:52	11:16	14:20	17:22	18:32
10	03:51	11:16	14:18	17:23	18:32	03:51	11:16	14:19	17:22	18:32
11	03:50	11:15	14:17	17:23	18:32	03:51	11:16	14:18	17:22	18:32
12	03:50	11:15	14:17	17:23	18:32	03:50	11:15	14:17	17:22	18:32
13	03:49	11:15	14:17	17:23	18:32	03:50	11:15	14:17	17:22	18:32
14	03:49	11:15	14:18	17:22	18:32	03:49	11:15	14:17	17:22	18:32
15	03:48	11:14	14:19	17:22	18:32	03:49	11:15	14:18	17:22	18:32
16	03:48	11:14	14:19	17:22	18:32	03:48	11:14	14:18	17:22	18:32
17	03:47	11:14	14:20	17:22	18:33	03:48	11:14	14:19	17:22	18:33
18	03:47	11:14	14:20	17:22	18:33	03:47	11:14	14:19	17:22	18:33

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
19	03:46	11:14	14:21	17:22	18:33	03:47	11:14	14:20	17:22	18:33
20	03:46	11:13	14:21	17:22	18:33	03:46	11:14	14:20	17:22	18:33
21	03:45	11:13	14:22	17:22	18:33	03:46	11:13	14:21	17:22	18:33
22	03:45	11:13	14:22	17:22	18:33	03:45	11:13	14:22	17:22	18:33
23	03:44	11:13	14:23	17:23	18:33	03:45	11:13	14:22	17:22	18:33
24	03:44	11:13	14:23	17:23	18:33	03:44	11:13	14:23	17:22	18:33
25	03:43	11:13	14:24	17:23	18:34	03:44	11:13	14:23	17:22	18:33
26	03:43	11:13	14:24	17:23	18:34	03:43	11:13	14:24	17:22	18:34
27	03:43	11:12	14:25	17:23	18:34	03:43	11:13	14:24	17:22	18:34
28	03:42	11:12	14:25	17:23	18:34	03:43	11:12	14:25	17:23	18:34
29	03:42	11:12	14:26	17:23	18:34	03:42	11:12	14:25	17:23	18:34
30	03:41	11:12	14:26	17:23	18:35	03:42	11:12	14:26	17:23	18:35
31	03:41	11:12	14:27	17:23	18:35	03:41	11:12	14:26	17:23	18:35

Tabel 4.4 Perbandingan jadwal waktu sholat kota Jakarta bulan Oktober 2011

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
1	04:20	11:42	14:48	17:47	18:56	04:21	11:43	14:49	17:47	18:56
2	04:19	11:42	14:47	17:47	18:56	04:20	11:42	14:48	17:47	18:56
3	04:19	11:41	14:46	17:46	18:56	04:20	11:42	14:48	17:47	18:56
4	04:18	11:41	14:45	17:46	18:56	04:19	11:42	14:47	17:47	18:56
5	04:18	11:41	14:44	17:46	18:55	04:19	11:41	14:46	17:46	18:56
6	04:17	11:40	14:43	17:46	18:55	04:18	11:41	14:45	17:46	18:56
7	04:17	11:40	14:42	17:46	18:55	04:18	11:41	14:44	17:46	18:56
8	04:16	11:40	14:42	17:46	18:55	04:17	11:41	14:43	17:46	18:56
9	04:16	11:39	14:42	17:46	18:55	04:17	11:40	14:42	17:46	18:56
10	04:15	11:39	14:41	17:46	18:55	04:16	11:40	14:41	17:46	18:56
11	04:15	11:39	14:41	17:46	18:55	04:16	11:40	14:41	17:46	18:56
12	04:14	11:39	14:42	17:45	18:55	04:15	11:39	14:42	17:46	18:56
13	04:14	11:38	14:42	17:45	18:55	04:14	11:39	14:43	17:46	18:55
14	04:13	11:38	14:43	17:45	18:55	04:14	11:39	14:43	17:46	18:55
15	04:13	11:38	14:43	17:45	18:55	04:14	11:39	14:44	17:45	18:56
16	04:12	11:38	14:44	17:45	18:55	04:13	11:38	14:44	17:45	18:56
17	04:12	11:38	14:44	17:45	18:55	04:13	11:38	14:45	17:45	18:56
18	04:11	11:37	14:45	17:45	18:55	04:12	11:38	14:45	17:45	18:56
19	04:11	11:37	14:45	17:45	18:55	04:12	11:38	14:46	17:45	18:56
20	04:10	11:37	14:46	17:45	18:55	04:11	11:38	14:46	17:45	18:56
21	04:10	11:37	14:46	17:45	18:56	04:11	11:37	14:47	17:45	18:56
22	04:09	11:37	14:47	17:45	18:56	04:10	11:37	14:47	17:45	18:56
23	04:09	11:36	14:47	17:45	18:56	04:10	11:37	14:47	17:45	18:56
24	04:09	11:36	14:48	17:45	18:56	04:09	11:37	14:48	17:45	18:56
25	04:08	11:36	14:48	17:45	18:56	04:09	11:37	14:48	17:45	18:56
26	04:08	11:36	14:49	17:45	18:56	04:09	11:37	14:49	17:45	18:56
27	04:07	11:36	14:49	17:45	18:56	04:08	11:37	14:49	17:45	18:57
28	04:07	11:36	14:50	17:45	18:57	04:08	11:36	14:50	17:45	18:57
29	04:07	11:36	14:50	17:45	18:57	04:07	11:36	14:50	17:46	18:57
30	04:06	11:36	14:51	17:46	18:57	04:07	11:36	14:51	17:46	18:57
31	04:06	11:36	14:51	17:46	18:57	04:07	11:36	14:51	17:46	18:57

Tabel 4.5 Perbandingan jadwal waktu sholat kota Denpasar bulan Oktober 2011

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
1	04:45	12:08	15:19	18:14	19:23	04:46	12:09	15:20	18:14	19:23
2	04:45	12:08	15:18	18:14	19:23	04:45	12:08	15:19	18:14	19:23
3	04:44	12:08	15:17	18:14	19:23	04:45	12:08	15:18	18:14	19:23
4	04:43	12:07	15:16	18:14	19:23	04:44	12:08	15:17	18:13	19:23
5	04:43	12:07	15:15	18:14	19:23	04:43	12:08	15:17	18:13	19:23
6	04:42	12:07	15:15	18:13	19:23	04:43	12:07	15:16	18:13	19:23
7	04:42	12:07	15:14	18:13	19:23	04:42	12:07	15:15	18:13	19:23
8	04:41	12:06	15:13	18:13	19:23	04:42	12:07	15:14	18:13	19:23
9	04:40	12:06	15:12	18:13	19:23	04:41	12:06	15:13	18:13	19:23
10	04:40	12:06	15:11	18:13	19:23	04:40	12:06	15:12	18:13	19:23
11	04:39	12:05	15:10	18:13	19:23	04:40	12:06	15:11	18:13	19:23
12	04:39	12:05	15:10	18:13	19:23	04:39	12:06	15:11	18:13	19:23
13	04:38	12:05	15:09	18:13	19:24	04:39	12:05	15:10	18:13	19:24
14	04:38	12:05	15:08	18:13	19:24	04:38	12:05	15:09	18:13	19:24
15	04:37	12:04	15:07	18:13	19:24	04:38	12:05	15:08	18:13	19:24
16	04:36	12:04	15:07	18:13	19:24	04:37	12:05	15:07	18:13	19:24
17	04:36	12:04	15:08	18:13	19:24	04:36	12:04	15:07	18:13	19:24
18	04:35	12:04	15:08	18:13	19:24	04:36	12:04	15:07	18:13	19:24
19	04:35	12:04	15:09	18:13	19:24	04:35	12:04	15:08	18:13	19:24
20	04:34	12:03	15:09	18:13	19:24	04:35	12:04	15:08	18:13	19:24
21	04:34	12:03	15:10	18:13	19:24	04:34	12:04	15:09	18:13	19:24
22	04:33	12:03	15:10	18:14	19:25	04:34	12:03	15:10	18:13	19:25
23	04:33	12:03	15:11	18:14	19:25	04:33	12:03	15:10	18:13	19:25
24	04:32	12:03	15:12	18:14	19:25	04:33	12:03	15:11	18:14	19:25
25	04:32	12:03	15:12	18:14	19:25	04:32	12:03	15:11	18:14	19:25
26	04:31	12:03	15:13	18:14	19:25	04:32	12:03	15:12	18:14	19:25
27	04:31	12:03	15:13	18:14	19:26	04:31	12:03	15:13	18:14	19:26
28	04:30	12:02	15:14	18:14	19:26	04:31	12:03	15:13	18:14	19:26
29	04:30	12:02	15:14	18:14	19:26	04:31	12:03	15:14	18:14	19:26
30	04:30	12:02	15:15	18:14	19:26	04:30	12:02	15:14	18:14	19:26
31	04:29	12:02	15:15	18:15	19:27	04:30	12:02	15:15	18:14	19:27

Tabel 4.6 Perbandingan jadwal waktu sholat kota Bekasi bulan Oktober 2011

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
1	04:19	11:41	14:48	17:47	18:55	04:20	11:42	14:49	17:46	18:55
2	04:19	11:41	14:47	17:47	18:55	04:19	11:41	14:48	17:46	18:55
3	04:18	11:41	14:46	17:46	18:55	04:19	11:41	14:47	17:46	18:55
4	04:18	11:40	14:45	17:46	18:55	04:18	11:41	14:46	17:46	18:55
5	04:17	11:40	14:44	17:46	18:55	04:18	11:40	14:45	17:46	18:55
6	04:17	11:40	14:43	17:46	18:55	04:17	11:40	14:44	17:45	18:55
7	04:16	11:39	14:42	17:46	18:55	04:17	11:40	14:43	17:45	18:55
8	04:16	11:39	14:41	17:46	18:55	04:16	11:40	14:42	17:45	18:55
9	04:15	11:39	14:40	17:46	18:55	04:16	11:39	14:41	17:45	18:55
10	04:14	11:39	14:41	17:46	18:55	04:15	11:39	14:40	17:45	18:55
11	04:14	11:38	14:41	17:46	18:55	04:15	11:39	14:40	17:45	18:55
12	04:13	11:38	14:42	17:45	18:55	04:14	11:38	14:41	17:45	18:55
13	04:13	11:38	14:42	17:45	18:55	04:13	11:38	14:41	17:45	18:55
14	04:12	11:38	14:43	17:45	18:55	04:13	11:38	14:42	17:45	18:55
15	04:12	11:37	14:43	17:45	18:55	04:13	11:38	14:42	17:45	18:55
16	04:12	11:37	14:44	17:45	18:55	04:12	11:38	14:43	17:45	18:55

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
17	04:12	11:37	14:44	17:45	18:55	04:12	11:37	14:43	17:44	18:55
18	04:11	11:37	14:45	17:45	18:55	04:11	11:37	14:44	17:44	18:55
19	04:11	11:37	14:45	17:45	18:55	04:11	11:37	14:44	17:44	18:55
20	04:10	11:36	14:46	17:45	18:55	04:10	11:37	14:45	17:44	18:55
21	04:09	11:36	14:46	17:45	18:55	04:10	11:36	14:45	17:44	18:55
22	04:09	11:36	14:47	17:45	18:55	04:09	11:36	14:46	17:44	18:55
23	04:08	11:36	14:47	17:45	18:55	04:09	11:36	14:46	17:44	18:55
24	04:08	11:36	14:48	17:45	18:55	04:08	11:36	14:47	17:44	18:55
25	04:08	11:36	14:48	17:45	18:55	04:08	11:36	14:47	17:44	18:55
26	04:07	11:36	14:49	17:45	18:56	04:08	11:36	14:48	17:45	18:56
27	04:07	11:35	14:49	17:45	18:56	04:07	11:36	14:48	17:45	18:56
28	04:06	11:35	14:49	17:45	18:56	04:07	11:36	14:49	17:45	18:56
29	04:06	11:35	14:50	17:45	18:56	04:06	11:35	14:49	17:45	18:56
30	04:06	11:35	14:50	17:46	18:56	04:06	11:35	14:50	17:45	18:56
31	04:05	11:35	14:51	17:46	18:57	04:06	11:35	14:50	17:45	18:57

Tabel 4.7 Perbandingan jadwal waktu sholat kota Pontianak bulan Oktober 2011

Tanggal	Penjadwalan oleh *Microcontroller*					Penjadwalan oleh PKPU				
	Subuh	Dhuhur	Azhar	Maghrib	Isya'	Subuh	Dhuhur	Azhar	Maghrib	Isya'
1	04:12	11:32	14:38	17:35	18:44	04:12	11:32	14:37	17:35	18:44
2	04:12	11:32	14:38	17:35	18:44	04:12	11:32	14:37	17:35	18:44
3	04:11	11:31	14:38	17:35	18:44	04:12	11:32	14:38	17:35	18:44
4	04:11	11:31	14:39	17:35	18:43	04:11	11:31	14:38	17:35	18:43
5	04:10	11:31	14:39	17:34	18:43	04:11	11:31	14:38	17:34	18:43
6	04:10	11:30	14:39	17:34	18:43	04:10	11:31	14:39	17:34	18:43
7	04:10	11:30	14:40	17:34	18:42	04:10	11:31	14:39	17:34	18:43
8	04:09	11:30	14:40	17:33	18:42	04:10	11:30	14:39	17:33	18:42
9	04:09	11:30	14:40	17:33	18:42	04:09	11:30	14:40	17:33	18:42
10	04:09	11:29	14:40	17:33	18:42	04:09	11:30	14:40	17:33	18:42
11	04:08	11:29	14:41	17:33	18:42	04:09	11:29	14:40	17:33	18:42
12	04:08	11:29	14:41	17:32	18:41	04:08	11:29	14:40	17:32	18:42
13	04:08	11:29	14:41	17:32	18:41	04:08	11:29	14:41	17:32	18:41
14	04:07	11:28	14:42	17:32	18:41	04:08	11:29	14:41	17:32	18:41
15	04:07	11:28	14:42	17:32	18:41	04:07	11:28	14:41	17:32	18:41
16	04:07	11:28	14:42	17:31	18:41	04:07	11:28	14:41	17:31	18:41
17	04:07	11:28	14:42	17:31	18:41	04:07	11:28	14:42	17:31	18:41
18	04:06	11:27	14:42	17:31	18:41	04:07	11:28	14:42	17:31	18:41
19	04:06	11:27	14:43	17:31	18:40	04:06	11:28	14:42	17:31	18:40
20	04:06	11:27	14:43	17:31	18:40	04:06	11:27	14:42	17:31	18:40
21	04:05	11:27	14:43	17:30	18:40	04:06	11:27	14:43	17:30	18:40
22	04:05	11:27	14:43	17:30	18:40	04:05	11:27	14:43	17:30	18:40
23	04:05	11:27	14:44	17:30	18:40	04:05	11:27	14:43	17:30	18:40
24	04:05	11:26	14:44	17:30	18:40	04:05	11:27	14:43	17:30	18:40
25	04:04	11:26	14:44	17:30	18:40	04:05	11:27	14:44	17:30	18:40
26	04:04	11:26	14:44	17:30	18:40	04:04	11:26	14:44	17:30	18:40
27	04:04	11:26	14:45	17:30	18:40	04:04	11:26	14:44	17:30	18:40
28	04:04	11:26	14:45	17:30	18:40	04:04	11:26	14:44	17:30	18:40
29	04:04	11:26	14:45	17:30	18:40	04:04	11:26	14:45	17:29	18:40
30	04:03	11:26	14:45	17:29	18:40	04:04	11:26	14:45	17:29	18:40
31	04:03	11:26	14:46	17:29	18:40	04:03	11:26	14:45	17:29	18:40

BAB V

PENUTUP

5.1. Simpulan

Dari penelitian ini dan dengan melihat masalah yang telah dirumuskan serta hasil pengujian, maka dapat diambil simpulan yaitu terdapat selisih pada perhitungan waktu sholat oleh *microcontroller* dengan jadwal sholat pada situs PKPU sebesar ± 3 menit. Keluaran informasi suara melalui ISD25120 dapat berjalan sesuai yang diharapkan, namun terdapat kekurangan yaitu adanya suara *background* yang mendengung.

5.2. Saran

Menurut hasil wawancara dengan Bapak Jimmy, kekurangan alat yang telah dibuat terdapat pada besar dimensinya yaitu 5.5x6.2x4.8 cm. Untuk mengatasi hal tersebut salah satu solusinya adalah melakukan penggantian komponen yang digunakan secara keseluruhan dengan komponen berkemasan SMT (*Surface Mount Technology*).

Kekurangan berikutnya adalah alat harus dioperasikan menggunakan *earphone*. Hal ini akan menyulitkan karena pengguna harus menggunakan *earphone* untuk mengetahui informasi waktu dan peringatan waktu sholat. Salah satu solusi yang ditawarkan adalah melakukan penambahan *speaker* internal, tentunya tanpa menambah besar dimensi alat.

Penggantian sumber catu daya baterai 9 V dengan sumber catu daya lain, seperti baterai *handphone*, akan membantu memperkecil dimensi alat dan

mengurangi biaya operasional alat karena baterai *handphone* bersifat *rechargeable*.

Terdapat kekurangan pada algoritma perhitungan dan penjadwalan waktu sholat, yaitu selisih sebesar ± 3 menit dari jadwal sholat yang terdapat pada situs PKPU. Maka dari itu perlu dicari algoritma perhitungan lain yang memiliki tingkat ketelitian lebih tinggi.

DAFTAR PUSTAKA

Anugraha, Rinto. 2009a. *Cara Menghitung Waktu Shalat*. (Online). (*http://www.eramuslim.com/syariah/ilmu-hisab/cara-menghitung-waktu-shalat.htm*). Diakses pada tanggal 2 Agustus 2011.

Anugraha, Rinto. 2009b. *Pengantar Ilmu Hisab*. (Online). (*http://www.eramuslim.com/syariah/ilmu-hisab/pengantar-ilmu-hisab.htm*). Diakses pada tanggal 2 Agustus 2011.

Anugraha, Rinto. 2009c. *Waktu-Waktu Shalat*. (Online). (*http://www.eramuslim.com/syariah/ilmu-hisab/waktu-waktu-shalat.htm*). Diakses pada tanggal 5 Agustus 2011.

Anugraha, Rinto. 2010. *Mengenal* Equation of Time. (Online). (*http://www.eramuslim.com/syariah/ilmu-hisab/mengenal-equation-of-time.htm*). Diakses pada tanggal 5 Agustus 2011.

ATMEL Corporation. 2011. 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash ATmega32 - ATmega32L. (Online). (*http://www.atmel.com/atmel/acrobat/doc2503.pdf*). Diakses pada tanggal 12 Juni 2011.

Bies, Lammert. 2011. RS232 Serial Connector Pin Assignment. (Online). (*http://www.lammertbies.nl/comm/cable/RS-232.html*). Diakses pada tanggal 14 Juni 2011.

Britain, Great. 1995. Astronomical Almanac for the Year 1996. United States: United States Government Printing.

Edwards. 2007. *Gerak Semu Matahari*. (Online). (*http://cosmicemission.wordpress.com/2007/08/05/gerak-semu-matahari/*). Diakses pada tanggal 3 Agustus 2011.

Faithtear. 2007. *Algoritma Konversi di Bidang Busur*. (Online). (*http://fathirhamdi.wordpress.com/2007/10/04/algoritma-konversi-di-bidang-busur/*). Diakses pada tanggal 13 Agustus 2011.

Husanto & Thomas. 2008. *Kupas Tuntas Mikrokontroler PIC16F84*. Yogyakarta : C.V ANDI OFFSET

Information Storage Device. 2000. ISD2500 Series. (Online). (*http://pdf1.alldatasheet.net/datasheet-pdf/view/143282/ETC1/ISD25120P/+0___9WVwSwJbKUHNCNzY/1+/datasheet.pdf*). Diakses pada tanggal 12 Juni 2011

INNOVATIVE ELECTRONICS. 2009. *AVR USB ISP*. (Online). (*http://www.innovativeelectronics.com/innovative_electronics/download_files/manual/Manual%20DT-HiQ%20AVR%20USB%20ISP.pdf*). Diakses pada tanggal 15 Juni 2011

Lohala. 2011. Simplex, Half Duplex, and Full Duplex. (Online). (*http://www.mystudyroom.com.np/classnotes.php?nan=89&fire=4&cake=26&sun=6&rnd=2387829974d3e5ee26be055.39465877*). Diakses pada tanggal 14 Juni 2011.

MAXIM Integrated Products. 2008. DS1307 64x8, Serial, I2C, Real-Time Clock. (Online). (*http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2688*). Diakses pada tanggal 12 Juni 2011.

Mazidi, M.A. 2000. The 8051 MICROCONTROLLER & Embedded System. New Jersey: Printice Hall.

Nalwan, P. A. 2003. *Panduan Praktis Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51*. Jakarta: PT Elex Media Komputindo.

Thomson, Jonathan. 2009. DIY USB to Serial Cable For $3. (Online). (*http://www.uchobby.com/index.php/2009/10/04/diy-usb-to-serial-cable-for-3/*). Diakses pada tanggal 25 Juni 2011.

Wicaksono, SP. 2004. *Menghitung Waktu Terbit dan Terbenam Matahari*. (Online). (http://wicax2.blogspot.com/2004_07_01_archive.html). Diakses pada tanggal 20 September 2011.

Winoto, Ardi. 2008. *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR*. Bandung : Informatika.