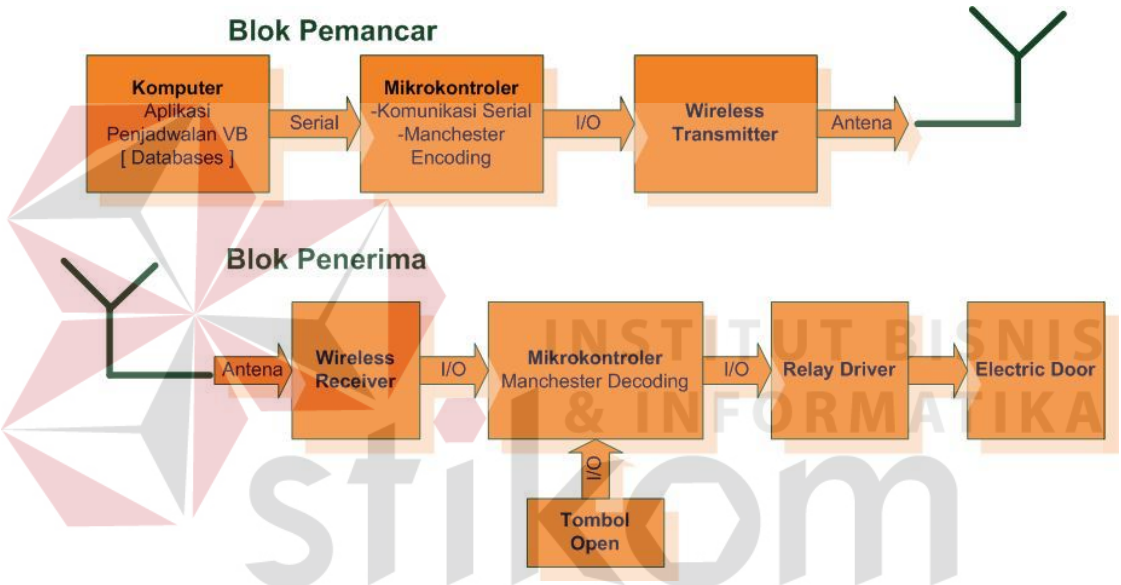
BAB III

METODE PENELITIAN

3.1 Model Penelitian

Penelitian yang dilakukan dapat dijelaskan dengan lebih baik melalui blok diagram seperti yang terlihat pada Gambar 3.1 dibawah ini :



Gambar 3.1 Blok Diagram

Pada Gambar 3.1 dapat dikelompokkan menjadi dua bagian utama, yaitu bagian pemancar (blok atas) serta bagian penerima (blok bawah).

1. Bagian Pemancar terdiri atas sebuah komputer *server*, mikrokontroler serta modul pemancar *Wireless*.
   a. Komputer merupakan *server* yang digunakan untuk mengatur penjadwalan tiap-tiap kelas. Pada komputer digunakan sebuah program *desktop* yaitu dengan menggunakan *software* Visual Basic. Aplikasi yang dibuat ini

digunakan untuk melakukan kontrol oleh penjaga serta digunakan untuk mengatur penjadwalan kelas. Seluruh informasi mengenai waktu, kelas, mata kuliah, dan lainnya disimpan dengan menggunakan *database*. Koneksi yang dibuat adalah koneksi *database* access dengan Visual Basic.

Program tersebut dapat berkomunikasi dengan mikrokontroler. Komunikasi yang digunakan adalah komunikasi *serial*. Dengan memanfaatkan komunikasi *serial* ini maka sebuah komputer dapat melakukan kontrol terhadap mikrokontroler.

b. Mikrokontroler digunakan untuk melakukan pengolahan data yang berasal dari komputer *server* sehingga dapat digunakan sebagai pengatur peralatan selanjutnya. Pada bagian ini sangat penting, karena mikrokontroler merupakan pengatur sehingga seluruh proses dapat berjalan sesuai dengan ketentuan yang diberikan.

   Selain hal tersebut, mikrokontroler inilah yang digunakan untuk menerapkan Metode Manchester sebagai *encoding* data ke modul *Wireless*, sehingga modul ini dapat berfungsi sebagaimana mestinya.

c. Modul *Wireless Transmitter* merupakan sebuah modul radio frekuensi 433 Mhz yang dapat mengirimkan data secara *Wireless* sehingga data *digital* tadi dapat diubah menjadi gelombang radio. Keunggulan dari modul ini adalah harga yang terjangkau (jika dibandingkan dengan modul RF lainnya), ukuran yang relatif kecil, serta dapat melakukan pengiriman data dengan baik.

2. Pada bagian penerima terdapat modul penerima *Wireless*, mikrokontroler, Tombol, *Relay driver*, serta *Electric Door Lock* yang berfungsi sebagai pengunci pintu.
   a. Modul *Wireless Receiver* berfungsi untuk menangkap gelombang radio (data analog) yang dikirimkan menjadi data *digital*. Data *digital* tersebut merupakan data yang dapat dipahami dan diolah oleh mikrokontroler untuk keperluan lebih lanjut.
   b. Mikrokontroler melakukan *decoding* dengan metode Manchester. Data yang telah diterjemahkan diolah menjadi perintah-perintah yang digunakan untuk menginstruksikan kapan pintu harus terkunci dan terbuka.
   c. Tombol *open* berfungsi sebagai tombol yang digunakan saat ingin membuka pintu. Tombol ini hanya akan berfungsi saat *flag* pintu terbuka diberikan oleh mikrokontroler, yang menandakan bahwa penjadwalan memberi perintah agar pintu terbuka.
   d. *Relay driver* digunakan agar dapat memicu *Electric Door Lock*. Arus dan tegangan yang tidak sesuai pada mikrokontroler mengharuskan pemakaian *relay driver*. *Relay driver* ini memberikan kebutuhan arus dan tegangan yang sesuai pada *Electric Door Lock*.
   e. Modul *Electric Door Lock* berfungsi sebagai pengunci pintu secara *auto*matis pada pintu. *Electric Door Lock* akan membuka disaat terdapat arus 1A dan tegangan 12V dan akan mengunci disaat tegangan bernilai 0V.

3.2 Cara Kerja Sistem Secara Keseluruhan

Aplikasi ini berfungsi sebagai pengunci pintu otomatis pada kelas disaat waktu telah menunjukan bahwa kelas tersebut dalam kegiatan belajar mengajar. Sistem terbagi menjadi 2 bagian yaitu pada sisi *server* (pemancar) dan *client* (penerima).

Server berlokasi di luar kelas yaitu pada petugas jaga di setiap lantai dan *client* berada pada kelas yang tepatnya di setiap pintu kelas. Tugas *server* adalah untuk menentukan kapan pintu kelas harus terbuka atau terkunci, sedangkan *client* hanya menerima perintah dari *server*.

Gambar 3.2 Ilustrasi Komputer

Pada sisi *server* terdapat sebuah komputer dengan program desktop Visual Basic 6.0 yang berfungsi untuk mengatur program penjadwalan. Tampilan depannya terlihat pada gambar 3.3. Penjelasan mengenai pemrograman Visual Basic akan dipaparkan pada sub bab berikutnya.

Gambar 3.3 Ilustrasi Program VB

Komputer *server* ini terhubung dengan sebuah mikrokontroler dengan menggunakan kabel serial agar dapat berkomunikasi dengan baik. Perintah-perintah yang diberikan oleh komputer diolah oleh mikrokontroler dan diubah menjadi data kode Manchester dengan menggunakan algoritma Manchester *encoding*. Yang kemudian diubah oleh modul TLP menjadi gelombang radio.

Gambar 3.4 Kabel Serial

Pada sisi penerima terdapat modul RLP yang berfungsi untuk mengubah gelombang radio menjadi data digital, yang saat ini masih berupa kode Manchester. Terdapat mikrokontroler dan *Electric Door Lock*, dimana mikrokontroler berfungsi sebagai penterjemah kode Manchester (algoritma Manchester *decoding*) hingga menjadi data asli.

Data asli tersebut segera diolah sehingga mikrokontroler dapat mengkontrol *Electric Door Lock* sesuai dengan printah yang diberikan oleh *server*. Mikrokontroler memiliki pin output dengan arus yang relatif rendah, oleh sebab itu dibutuhkan serangkaian *relay driver* untuk mengatasi hal tersebut.

Gambar 3.5 *Electric Door*

Sistem penguncian yang dilakukan pada sisi penerima yaitu *Electric Door Lock* akan diaktifkan jika terdapat penekanan tombol dan *server* telah memberi perintah untuk membuka pintu. Hal tersebut dilakukan karena secara default *Electric Door Lock* bersifat mengunci dan saat mengaktifkan *Electric Door Lock* hanya diperbolehkan selama 20 detik. Oleh karena hal diatas maka *Electric Door Lock* akan membuka saat dibutuhkan saja (saat tombol ditekan) dan akan

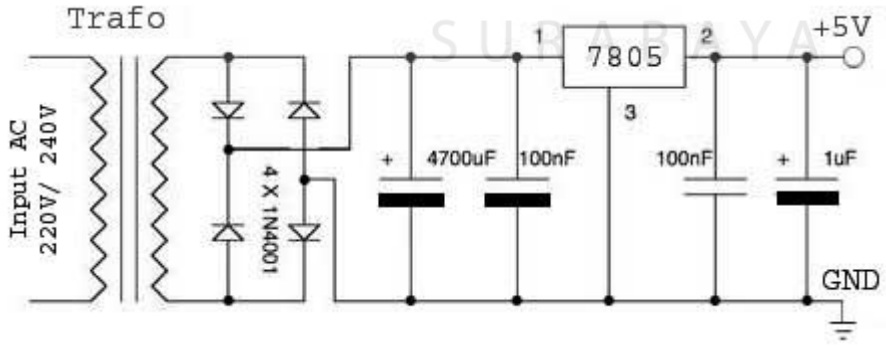
mengkunci secara otomatis kurang dari 20 detik agar *Electric Door Lock* tidak rusak.

3.3 Perancangan *Hardware*

Peralatan ini membutuhkan berbagai macam rangkaian *hardware* agar dapat menjalankan fungsinya dengan baik. Penjelasan mengenai perancangan *hardware* ini terbagi menjadi beberapa bagian, yang diantaranya : rangkaian *power supply*, rangkaian sistem minimum, rangkaian tombol, rangkaian *serial*, rangkaian *relay driver*, dan rangkaian modul *wireless*.

3.3.1 Rangkaian *Power supply*

Rangkaian *power supply* merupakan rangkaian yang berfungsi sebagai pengubah tegangan AC menjadi tegangan DC serta dapat memberikan kebutuhan daya pada rangkaian.



Gambar 3.6 Rangkaian *Power supply* 5Volt

Cara kerja rangkaian

Tegangan murni AC 220 V/ 240V dari PLN diturunkan oleh *Trans*form*ator* (Trafo) yang mempunyai fungsi untuk menurunkan dan menaikan tegangan AC.

Dalam hal ini tegangan sudah diturunkan menjadi 12 Volt AC. Tegangn 12VAC ini kemudian disearahkan dengan 4 buah Dioda (Dioda *Bridge*) 1N4001 menjadi tegangan searah 12 volt s/d 16 Volt.

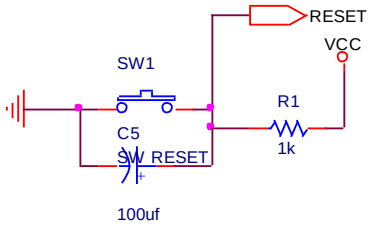
Tegangan DC tersebut belum benar-benar DC tetapi masih terdapat *ripple* AC dengan frekuensi sesuai *input* AC dari PLN (50-60 Hz). Di sinilah fungsi dua buah *Condensator* 4700uF dan 100nF yang bertugas menyaring dan memperkecil *ripple* AC sehingga makin mendekati grafik tegangan DC.

Hasil saringan tersebut masih belum mencapai tegangan yang diinginkan (5 Volt), untuk itu diperlukan IC *regulator* 7805 yang berfungsi untuk menstabilkan tegangan output menjasi 5 Volt DC. *Ripple* AC yang masih ada di buang melalui dua *Condensator* 100nF dan 1uF.

Jadi tegangan AC (bolak-balik) 220V/220V AC dari PLN, setelah diproses melalui rangkaian *regulator* DC 5Volt ini akan menjadi tegangan stabil DC (searah) 5 Volt yang dapat digunakan sebagai *power supply* DC perangkat elektronik yang sesuai.(http://almarwah.sch.id/*regulator*-tegangan-5-volt/, diakses September 2011)

3.3.2 Rangkaian Sistem Minimum

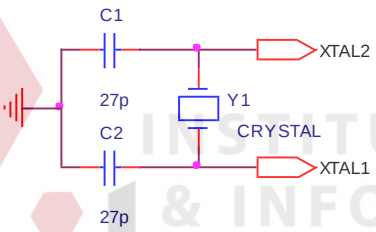
Sistem minimum mikrokontroler adalah rangkaian elektronik minimum yang diperlukan untuk beroperasinya IC mikrokontroler. Rangkaian sistem minimum terbagi menjadi 2 rangkaian utama yaitu rangkaian *reset* dan rangkaian *crystal*.



Gambar 3.7 Rangkaian *Reset*

Cara Kerja Rangkaian :

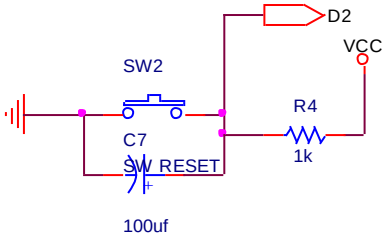
Rangkaian R-C pada tombol *reset* (PB1) digunakan untuk mengurangi *noise* serta memiliki fungsi terpenting yaitu untuk melakukan *reset* saat pertama kali catu daya dinyalakan. *Reset* untuk pertama kali merupakan hal yang terpenting sehingga dapat memastikan bahwa program telah berada pada posisi awal.



Gambar 3.8 Rangkaian *Crystal*

3.3.3 Rangkaian Tombol

Tombol merupakan komponen yang kerap digunakan pada setiap projek. Setiap tombol pasti memiliki *noise* yang dapat menyebabkan pembacaan pada mikrokontroler sedikit terganggu, oleh sebab itu digunakanlah rangkaian *filter* R-C untuk mengatasi hal tersebut.



Gambar 3.9 Rangkaian Tombol

Rangkaian *filter* R-C merupakan rangkaian yang berfungsi sebagai pemotong frekuensi. Dengan demikian *noise* yang pada umumnya terletak pada tepi atas gelombang dapat dipotong sehingga *noise* akan berkurang bahkan hilang. Rangkaian ini memiliki rumus :

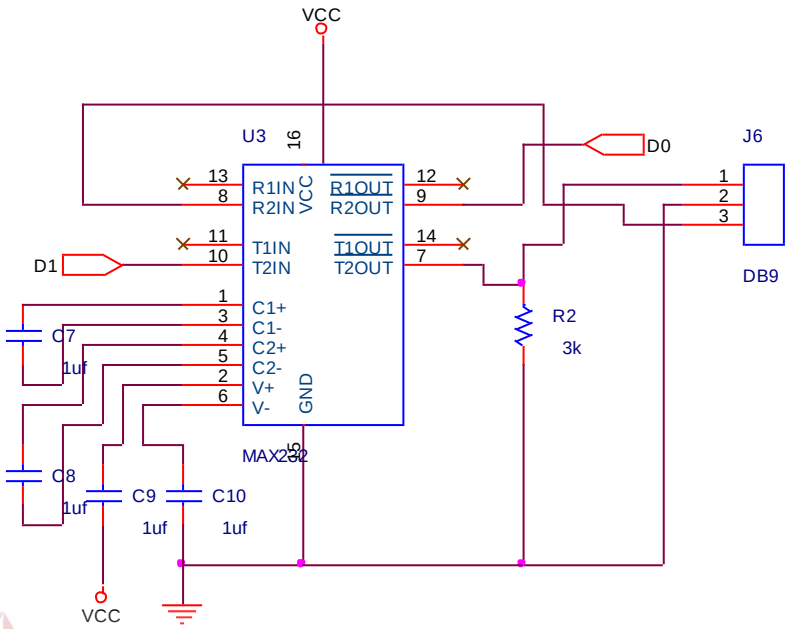
$$f = \frac{1}{2\pi rc}$$

Gambar 3.10 Rumus Frekuensi *Cut off*

Sehingga pada rangkaian pada gambar 3.10 frekuensi yang dapat dipotong oleh rangkaian ini sebesar 1,6 Hz.

3.3.4 Rangkaian *Serial*

Rangkaian *serial* merupakan rangkaian yang dibutuhkan agar sebuah mikrokontroler dapat berkomunikasi secara *serial* dengan komputer atau peralatan lain. Mikrokontroler menggunakan TTL sebagai *input* dan *output* data, yang berbeda dengan komputer personal. Oleh sebab itu dibutuhkan sebuah rangkaian yang dapat digunakan untuk menjebatani hal tersebut. Pada gambar 3.11 terlihat rangkaian *serial* dengan menggunakan IC MAX232.



Gambar 3.11 Rangkaian *Serial*

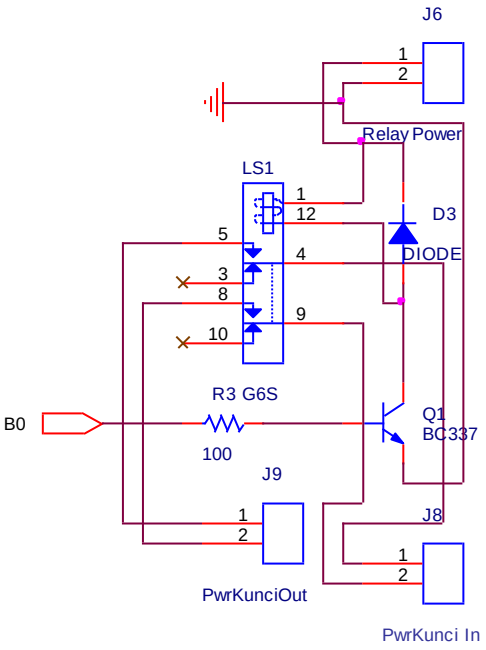
Penjelasan Rangkaian :

Pada gambar 3.11 terlihat bahwa antara Rx dan Tx dibuat *crozz* (terbalik), T*in* masuk pada Tx Mikro dan Rin masuk pada DB9 (*interface serial* ke komputer) serta Tout masuk pada DB9 dan R*out* Menuju Rx Mikro.

Terdapat 4 buah kapasitor dan sebuah resistor yang digunakan untuk memperhalus data yang masuk serta mengurangi *noise*, dimana nilai-nilai yang ditentukan disesuaikan dengan data sheet pada IC MAX232.

3.3.5 Rangkaian *Relay driver*

Relay merupakan komponen yang memungkinkan sebuah rangkaian dengan daya rendah agar dapat melakukan *switch on* dan *off* dengan arus yang relative tinggi.



Gambar 3.12 Rangkaian *Relay driver*

Penjelasan Komponen Rangkaian :

Relay yang digunakan adalah *relay* G6S buatan Omron dengan spesifikasi tegangan 5Volt, Arus 28,1mA, dan resistansi *coil* sebesar 178 Ohm. Untuk cara penghitungan telah dijelaskan pada bab sebelumnya, dimana cara perhitungan untuk *relay driver* adalah sebagai berikut : Arus yang dibutuhkan pada basis sebesar 28,1mA / 100 (hFE BC337 sebesar 100 untuk 100mA) sehingga minimal arus yang diberikan sebesar 0,281mA namun pada prakteknya tidaklah demikian. Oleh karenanya arus yang disediakan harus sebesar 2 x 0,281mA = 0,562mA.

Sehingga nilai resistansi pada R3 agar dapat memenuhi 0,562mA pada tegangan 5Volt adalah sebesar 8,89 Kohm. Berhubung dipasaran tidak memungkinkan untuk mendapatkan resistor dengan nilai yang tepat 8,89 Kohm maka digunakanlah resistor sebesar 9,1 Kohm.

Pemasangan dioda digunakan untuk mencegah tegangan balik yang di akibatkan oleh *coil* saat *on* dan *off*, sehingga digunakan sebuah diode sebagai

pengaman. Konektor *relay power* digunakan sebagai pemicu tegangan agar dapat menjalankan *relay*, sedangkan konektor PwrKunciIn dan PwrKunciOut merupakan saklar yang digunakan untuk memberi daya pada alat pengunci pintu. Berikut ini adalah gambaran dari alat pengunci pintu dengan berbasis *solenoid* yang terlihat pada gambar 3.13.

Gambar 3.13 Pengunci Pintu

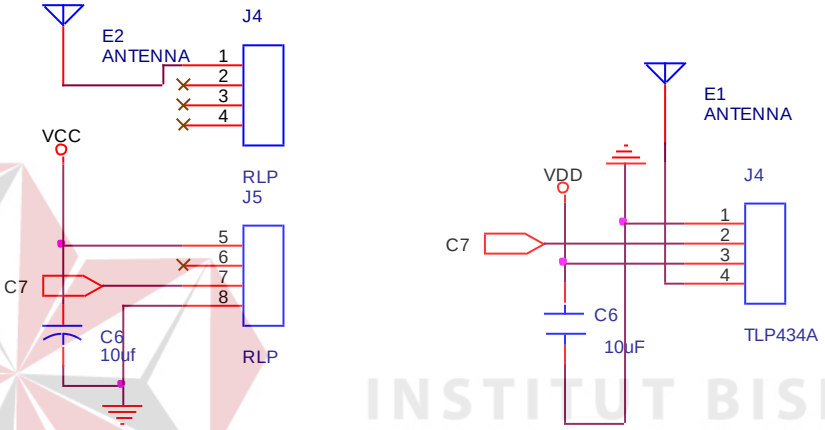
Saat *relay driver* diberi tegangan 5V maka kedua konektor PwrKunci akan terhubung sehingga dapat menyalakan alat pengunci pintu, sedangkan saat tegangan bernilai 0V maka kedua konektor tidak akan tersambung.

3.3.6 Rangkaian Modul *Wireless*

Modul *wireless* berfungsi sebagai pengirim data berupa gelombang radio dengan media udara. Terdapat dua buah modul yaitu modul pemancar dan modul penerima. Gambar 3.14 merupakan modul *wireless* RLP & TLP 433.

Gambar 3.14 RLP & TLP 433

Untuk dapat berfungsi dengan baik maka modul ini membutuhkan sedikit komponen tambahan yang terlihat pada gambar 3.15.



Gambar 3.15 Rangkaian Modul *Wireless*

Penjelasan Rangkaian :

Secara garis besar, komponen yang diberikan untuk kedua modul ini tidaklah berbeda. Terlihat bahwa untuk dapat mengoperasikan modul ini hanya dibutuhkan sebuah kapasitor sebagai kopling saja. Antena yang dibutuhkan juga

3.4 Perancangan Aplikasi Visual Basic

Aplikasi Visual Basic digunakan untuk melakukan pengolahan waktu penjadwalan untuk setiap ruang kelas. Pada aplikasi ini digunakan *database* guna dapat melakukan penyimpanan waktu yang telah ditetapkan oleh pengguna.

Database yang digunakan hanya menggunakan sebuah tabel seperti yang terlihat pada Tabel 3.1.

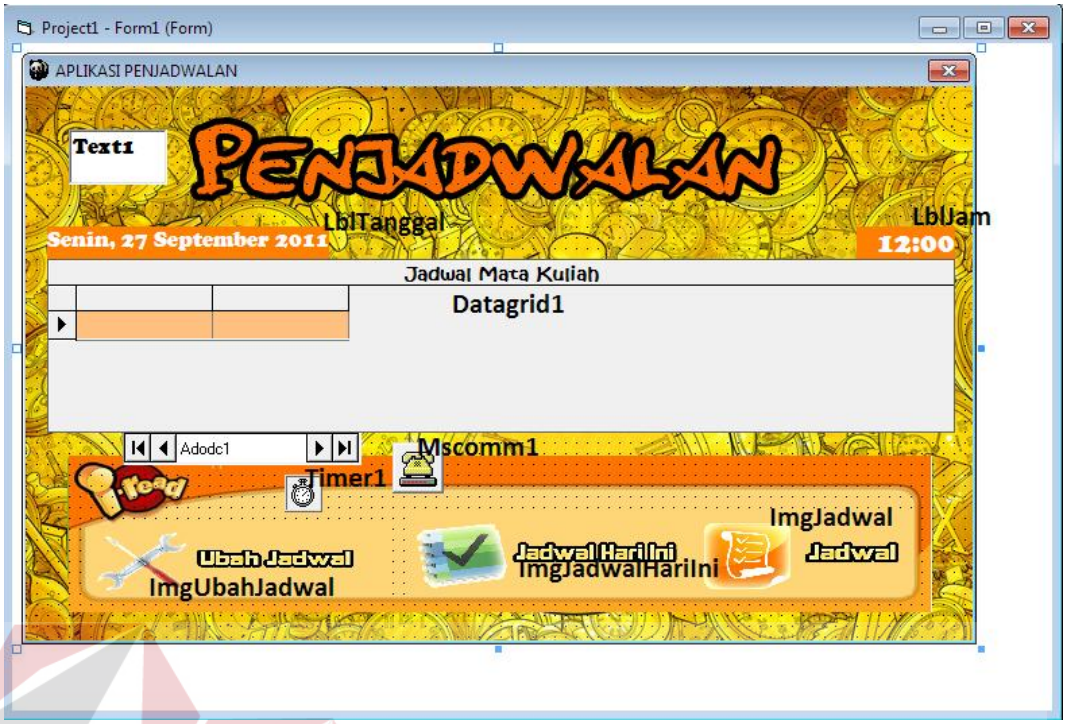
Tabel 3.1 *Database* Access

KodeMK	NamaMK	Ruang	Hari	Jam Mulai	Menit Mulai	Jam Selesai	Menit Selesai	StatusPintu	Dosen	Terlambat
1001	Algoritma Pemrograman	B301	Sunday	13	50	8	0	0	Harianto	0
1002	Kalkulus	B301	Monday	6	0	7	30	1	Ira	2
1003	Basis Data	B302	Monday	7	0	9	30	0	Rini	1
1004	Fisika	B301	Tuesday	8	40	10	0	0	Ira	5
1005	Jaringan Komputer	B301	Thursday	10	35	13	30	0	Dewa	2
1006	Mikrokontroler Dasar	B302	Wednesday	12	30	15	30	0	Iik	0
1007	Mikrokontroler Lanjut	B302	Friday	8	30	11	50	1	Iik	0

Secara garis besar penjelasan mengenai aplikasi penjadwalan ini dapat dibagi menjadi dua bagian utama yaitu desain dan penggunaan form dan program yang digunakan.

3.4.1 Desain dan Kegunaan Form

Perangkat lunak dari alat penguncian pintu otomatis dibuat menggunakan Visual Basic 6.0, dengan tampilan seperti gambar --. Sebagai antarmuka *software* dengan alat maka digunakan program Visual Basic yang aturan penulisannya didasari dengan aturan penulisan bahasa basic. Aplikasi Visual Basic ini terbagi menjadi 2 buah form yaitu form aplikasi penjadwalan (pada gambar 3.16) dan form ubah jadwal (pada gambar 3.17) dari sisi programmer.

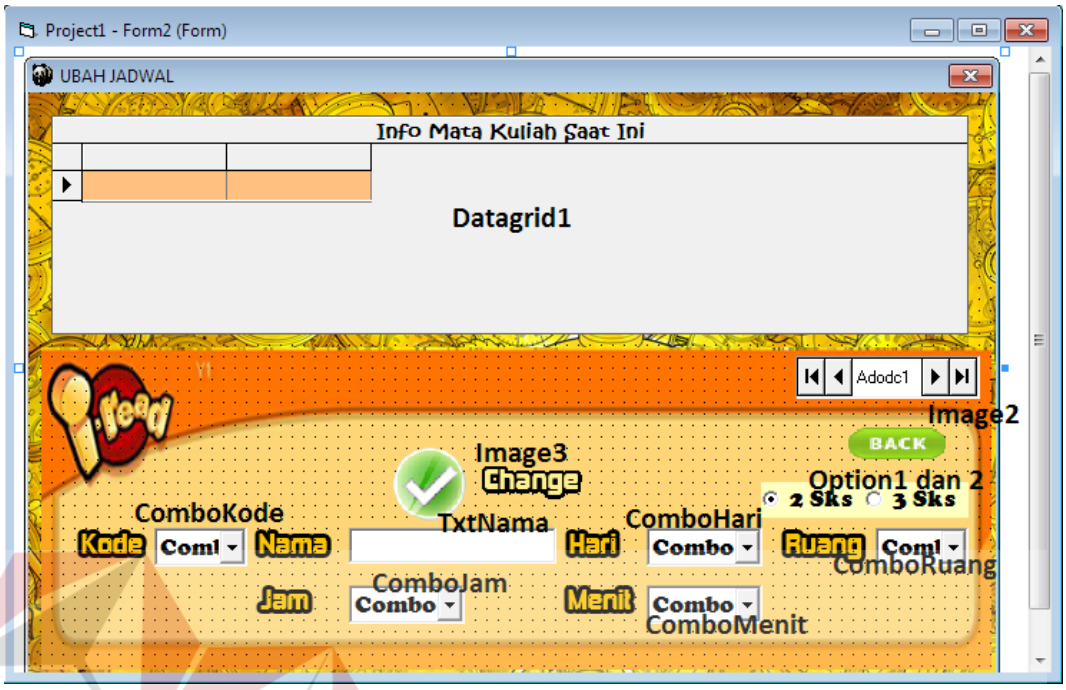


Gambar 3.16 Form *Main* Menu

Keterangan Gambar :

- LblTanggal : Menampilkan hari dan tanggal saat program dijalankan.
- LblJam : Menampilkan waktu saat program dijalankan.
- Datagrid1 : Menampilkan jadwal pada *database*.
- Adodc1 : Mengkoneksikan *database* access.
- Mscomm1 : Memberikan akses komunikasi serial pada aplikasi.
- *Timer*1 : Melakukan *refresh* waktu dan proses pengiriman serial.
- ImgUbahJadwal : Membuka form ubah jadwal.
- ImgJadwalHariIni : Mengubah datagrid1 menjadi jadwal hari ini.
- ImgJadwal : Mengubah datagrid1 menjadi jadwal lengkap.

Pada form kedua (form ubah jadwal) secara umum berfungsi sebagai pengubah jadwal yang tersimpan dalam *database*. Gambar 3.17 merupakan penampakan form ubah jadwal dari sisi programmer.



Gambar 3.17 Form Menu Kedua

Keterangan Gambar :

- Datagrid1 : Menampilkan jadwal pada *database*.
- Adodc1 : Mengkoneksikan *database* access.
- Image2 : Menutup form dan kembali ke form utama.
- Image3 : Melakukan *update database* sesuai inputan.
- *Combo*kode : Menampilkan kode mata kuliah yang tersedia.
- TxtNama : Sebagai input dalam melakukan perubahan nama.
- *Combo*Jam : Memberi pilihan jam (1 - 24).
- *Combo*Menit : Memberi pilihan menit (0 - 59).
- *Combo*Hari : Memberi pilihan hari (monday - sunday).
- *Combo*Ruang : Memberi pilihan ruang kelas (B301-B302).
- Option1 : Mengatur lama perkuliahan sebesar 100 menit.

- Option2 : Mengatur lama perkuliahan sebesar 150 menit.

3.4.2 *Programming* VB6

Pembuatan program dengan menggunakan Visual Basic terbagi menjadi dua bagian yaitu pengaturan properties dan penulisan koding. Berikut ini merupakan penjelasan mengenai pengaturan properties pada setiap *object*.

Gambar 3.18 Form Menu Ketiga

Keterangan Propertis :

Tabel 3.2 Propertis *Main* Menu

NAMA OBJECT	PROPERTIS	VALUE	NAMA OBJECT	PROPERTIS	VALUE
LblTanggal	Caption	Senin, 27 September 2011	Mscomm1	CommPort	1
	Backcolor	&H000080FF&		InputLen	1
	Font	Arial		Rthreshold	1
LblJam	Caption	12:00		Setting	9600,n,8,1
	Backcolor	&H000080FF&	Timer1	Enable	TRUE
	Font	Arial		Interval	50000
Datagrid1	Apparance	1-dbg3D	ImgUbahJadwal1	Visible	TRUE
	AllowAddNew	FALSE	ImgUbahJadwal2	ToolTipText	Untuk Set Ulang Jadwal
	AllowArrow	FALSE		Visible	FALSE
	AllowDelete	FALSE	ImgJadwalHariIni1	Visible	TRUE
	AllowUpdate	FALSE	ImgJadwalHariIni2	ToolTipText	Jadwal Sekarang
	Backcolor	&H0080C0FF&		Visible	FALSE
	Caption	Jadwal Mata Kuliah	ImgJadwal1	Visible	TRUE
	Font	Arial	ImgJadwal2	ToolTipText	Lihat Jadwal
	HeadFont	Arial		Visible	FALSE
Adodc1	CommandType	1-adCmdText			
	ConnectionStrin	Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Penjadwalan.mdb;Persist Security Info=False			
	RecordSource	SELECT * FROM Kelas			

Pada setiap pemrograman dibutuhkan beberapa variabel guna membantu dalam pembuatan aplikasi. Berikut ini merupakan deklarasi variabel pada form utama (form aplikasi penjadwalan).

```
Dim SB301 As Integer
Dim SB302 As Integer
```

Form load merupakan program yang akan dijalankan disaat from pertama kali dibuka. Pada form load secara garis besar berisi mengenai nilai awal dan sebagai proses inisialisasi seluruh *object* yang dibutuhkan.

```
Private Sub Form_Load()
```

```
'Load DataGrid Jadwal Hari Ini
Dim hari As String
hari = WeekdayName(Weekday(Now))
Adodc1.RecordSource = "SELECT * FROM Kelas WHERE Hari =" & "'" & hari &
"'"
Adodc1.Refresh
'------------------------------

'Menuliskan Hari dan Tanggal pada Label
LblTanggal.Caption = hari & " " & DateValue(Now)
LblJam.Caption = Hour(Now) & " : " & Minute(Now)
'--------------------------------------

'Membuka Port Serial
MSComm1.PortOpen = True
'-------------------

End Sub
```

Pada aplikasi, setiap image yang terkena mouse akan nampak sebuah animasi yang membuat seakan-akan image tersebut adalah tombol yang akan menyala saat terkena mause.

Gambar 3.19 Animasi Tombol

Berikut ini adalah cuplikan *syntax* untuk pembuatan animasi tersebut.

```
'Reset Animasi Warna
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
ImgUbahJadwal2.Visible = False
ImgUbahJadwal.Visible = True
ImgJadwalHariIni2.Visible = False
ImgJadwalHariIni.Visible = True
ImgJadwal2.Visible = False
ImgJadwal.Visible = True

End Sub
'-------------

'Animasi Warna
Private Sub ImgJadwal_mousemove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
ImgUbahJadwal2.Visible = False
ImgUbahJadwal.Visible = True
ImgJadwalHariIni2.Visible = False
ImgJadwalHariIni.Visible = True
ImgJadwal2.Visible = True
ImgJadwal.Visible = False
End Sub
```

```
'-------------
'Animasi Warna
Private Sub ImgJadwalHariIni_mousemove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
ImgUbahJadwal2.Visible = False
ImgUbahJadwal.Visible = True
ImgJadwalHariIni2.Visible = True
ImgJadwalHariIni.Visible = False
ImgJadwal2.Visible = False
ImgJadwal.Visible = True
End Sub


'-------------

'Animasi Warna
Private Sub ImgUbahJadwal_MouseMove(Button As Integer, Shift As Integer,
X As Single, Y As Single)
ImgUbahJadwal2.Visible = True
ImgUbahJadwal.Visible = False
ImgJadwalHariIni2.Visible = False
ImgJadwalHariIni.Visible = True
ImgJadwal2.Visible = False
ImgJadwal.Visible = True
End Sub

'-------------
```

Ada sebauh *textbox* yang seakan-akan tidak berfungsi yaitu text1, namun fungsi text1 ini sebenarnya adalah untuk menampung pembacaan pada komunikasi serial sehingga programmer dapat memeriksa apakah komunikasi berjalan dengan baik atau tidak.

Untuk memunculkan text1 saat program berjalan, dibutuhkan trik khusus yaitu dengan melakukan double klik pada label tanggal. Berikut ini adalah rahasia *syntax* tersebut.

```
Private Sub LblTanggal_DblClick()
Text1.Visible = True
End Sub
```

Saat menggunakan mscomm maka saat program telah selesai digunakan atau ditutup oleh user maka port serial yang tadinya terbuka harus ditutup kembali.

Gambar 3.20 *Object* Mscomm

Tujuannya agar aplikasi lain dapat memanfaatkan port serial ini karena jika port tetap dibiarkan terbuka maka aplikasi lain menganggap bahwa port serial masih digunakan. Berikut merupakan *syntax* untuk menutup port serial tersebut.

```
Private Sub Form_Unload(Cancel As Integer)
MSComm1.PortOpen = False
       End Sub
```

Mscomm digunakan sebagai alat pada Visual Basic untuk dapat mengakses port serial pada komputer. Berikut ini merupakan *syntax* untuk membaca data yang dikirimkan oleh mikrokontroler kepada komputer.

```
Private Sub MSComm1_OnComm()
If MSComm1.InBufferCount <> 0 Then
    Text1.Text = MSComm1.Input
End If
End Sub
```

Untuk dapat mengubah tampilan pada data grid view dibutuhkan kemampuan dalam melakukan coding SQL guna mengubah-ubah query yang ingin ditampilkan.

Gambar 3.21 Tombol *Main* Menu

Berikut ini adalah *syntax* yang digunakan untuk mengubah tampilan grid view dengan mengakses adodc1.

```
'Perintah untuk lihat Jadwal Lengkap
Private Sub ImgJadwal2_Click()
Adodc1.RecordSource = "SELECT * FROM Kelas "
Adodc1.Refresh
End Sub
```

```
'-----------------------------------


'Perintah untuk lihat Jadwal Hari Ini
Private Sub ImgJadwalHariIni2_Click()
Dim hari As String
hari = WeekdayName(Weekday(Now))
Adodc1.RecordSource = "SELECT * FROM Kelas WHERE Hari =" & "'" & hari &
"'"
Adodc1.Refresh
End Sub


'-------------------------------------
```

Tombol image yang terakhir adalah tombol ubah jadwal. Secara umum coding yang dilakukan pada tombol ini adalah memanggil form kedua dan menutup form yang bersangkutan.

```
'Perintah untuk Ubah Jadwal
Private Sub ImgUbahJadwal2_Click()
Load Form2
Form2.Show
Unload Me
End Sub

'--------------------------
```

Poin utama yang sangat penting pada form *main* menu ini terdapat pada *object* yang terakhir ini yaitu *timer*1. *Timer* 1 berfungsi untuk meng*update* tanggal, waktu, melakukan proses kalkulasi kapan status pintu terbuka atau tertutup, menentukan pengiriman data serial, dan mengirim kondisi sesuai dengan aturan berkomunikasi dengan mikrokontroler. Terlihat pada cuplikan *syntax* program dibawah ini.

```
Private Sub Timer1_Timer()
    'Update Label Waktu
    LblJam.Caption = Hour(Now) & " : " & Minute(Now)
    '-----------------------------------------------------

    Dim hitmenit, hitmenitMulai, hitmenitSelesai As Integer

    hitmenit = Hour(Now) * 60 + Minute(Now)
    For i = 0 To Adodc1.Recordset.RecordCount - 1

        'Update Status Pintu
        hitmenitMulai   =   (Adodc1.Recordset.Fields!JamMulai   *   60)   +
                        Adodc1.Recordset.Fields!MenitMulai
```

```
        hitmenitSelesai  =  (Adodc1.Recordset.Fields!JamSelesai  *  60)  +
                        Adodc1.Recordset.Fields!MenitSelesai
        If (hitmenitMulai <= hitmenit) And (hitmenitSelesai > hitmenit)
        Then
            Adodc1.Recordset.Fields!StatusPintu = "1"
        Else
            Adodc1.Recordset.Fields!StatusPintu = "0"
        End If
        '-----------------------------------------------

        'Menentukan Pengiriman Data Serial
        If     Adodc1.Recordset.Fields!StatusPintu     =     "1"     And
        Adodc1.Recordset.Fields!Ruang = "B301" Then
            SB301 = 1
        ElseIf    Adodc1.Recordset.Fields!StatusPintu    =    "0"    And
        Adodc1.Recordset.Fields!Ruang = "B301" Then
            SB301 = 0
        End If
        If     Adodc1.Recordset.Fields!StatusPintu     =     "1"     And
        Adodc1.Recordset.Fields!Ruang = "B302" Then
            SB302 = 1
        ElseIf    Adodc1.Recordset.Fields!StatusPintu    =    "0"    And
        Adodc1.Recordset.Fields!Ruang = "B302" Then
            SB302 = 0
        End If
        '-------------------------------------------------------


        Adodc1.Recordset.MoveNext
    Next
    Adodc1.Recordset.MoveFirst

'Mengirim Data Serial
    If SB301 = 0 And SB302 = 0 Then
        MSComm1.Output = "A"
    ElseIf SB301 = 1 And SB302 = 0 Then
        MSComm1.Output = "B"
    ElseIf SB301 = 0 And SB302 = 1 Then
        MSComm1.Output = "C"
    ElseIf SB301 = 1 And SB302 = 1 Then
        MSComm1.Output = "D"
    End If
End Sub
```

Gambar 3.22 Menu Keterlambatan

Pada program Visual Basic ini terdapat fasilitas tersembunyi untuk melakukan pencatatan jika terjadi keterlambatan pada dosen. Menu rahasia akan muncul disaat user melakukan klik dua kali pada label jam. Saat image rahasia ini muncul maka *timer* akan dimatikan hingga user melakukan klik dua kali lagi.

```
Private Sub LblJam_DblClick()
If statusRahasia = 0 Then
    statusRahasia = 1
    Image1.Visible = True
    Timer1.Enabled = False
Else
    statusRahasia = 0
    Image1.Visible = False
    Timer1.Enabled = True
End If
```

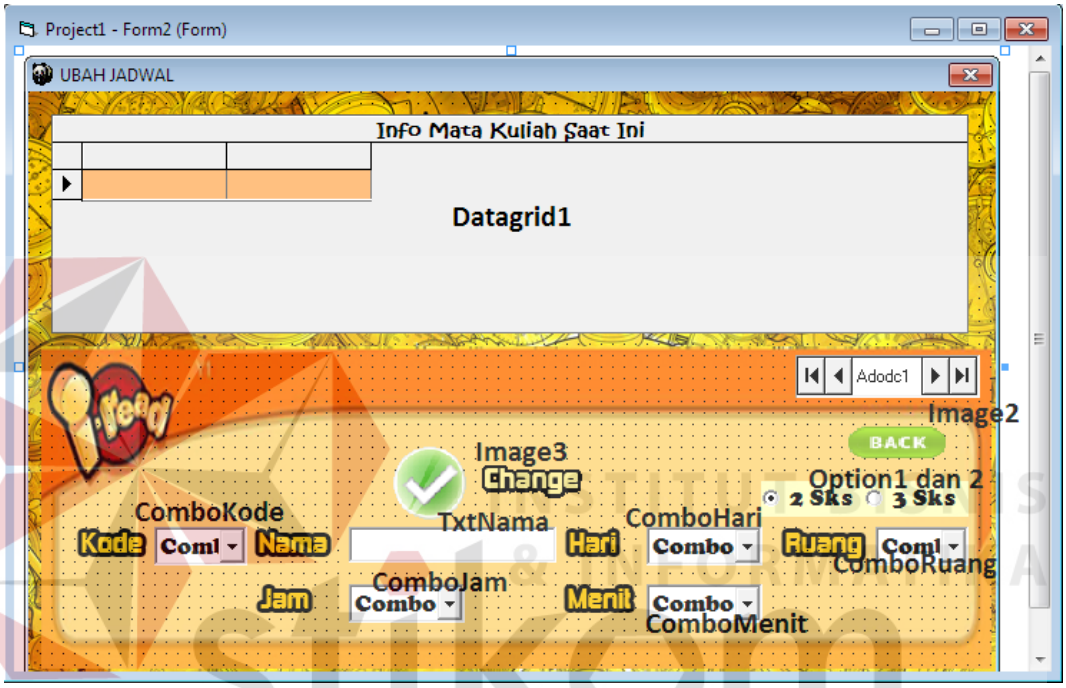
Saat image rahasia telah muncul maka user dapat melakukan klik yang akan berdampak menambahkan jumlah keterlambatan pada dosen yang bersangkutan. Program akan memerintahkan mikrokontroler dengan mengirimkan data serial untuk membuka pintu.

```
Private Sub Image1_Click()
    Adodc1.Recordset.Fields!Terlambat=Adodc1.Recordset.Fields!Terlambat+1
    Adodc1.Recordset.Fields!StatusPintu = 1
```

```
    MSComm1.Output = "D"

    MsgBox "Keterlambatan telah diTambahkan", vbOKOnly, "Admin"
End Sub
```

Pada uraian diatas telah dijelaskan mengenai pemrograman yang dilakukan pada form utama, selanjutnya mengenai uraian form kedua yaitu form ubah jadwal yang terlihat pada gambar 3.23.



Gambar 3.23 Form Menu Kedua

Keterangan Propertis :

Tabel 3.3 Propertis Form Kedua

NAMA OBJECT	PROPERTIS	VALUE	NAMA OBJECT	PROPERTIS	VALUE
Datagrid1	Apparance	1-dbg3D	ComboHari	Font	Cooper Black
	AllowAddNew	FALSE		Style	2-Dropdown List
	AllowArrow	FALSE	ComboRuang	Font	Cooper Black
	AllowDelete	FALSE		Style	2-Dropdown List
	AllowUpdate	FALSE	Option1	Backcolor	&H00C0FFFF&
	Backcolor	&H0080C0FF&		Font	Cooper Black
	Caption	Jadwal Mata Kuliah	Option2	Backcolor	&H00C0FFFF&
	Font	Arial		Font	Cooper Black
	HeadFont	Arial	Image1	Visible	TRUE
Adodc1	CommandType	1-adCmdText	Image3	ToolTipText	Ubah Data
	ConnectionString	Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Penjadwalan.mdb;Persist Security Info=False		Visible	FALSE
	RecordSource	SELECT * FROM Kelas	Image2	ToolTipText	Kembali ke Main Menu
ComboKode	Font	Cooper Black		Visible	TRUE
	Style	2-Dropdown List			
ComboJam	Font	Cooper Black			
	Style	2-Dropdown List			
ComboMenit	Font	Cooper Black			
	Style	2-Dropdown List			

Seperti uraian sebelumnya dikatakan bahwa setiap program yang akan dijalankan membutuhkan proses untuk mengisi nilai awal dan inisialisasi *object*. Berikut ini merupakan proses inisialisasi yang dilakukan pada form kedua.

```
'Proses inisialisasi
Private Sub Form_Load()
Dim i As Integer
Adodc1.RecordSource = "SELECT * FROM Kelas "
Adodc1.Refresh
'Memasukan kode MK pada Combokode
For i = 0 To Adodc1.Recordset.RecordCount - 1
    Combokode.AddItem Adodc1.Recordset.Fields!KodeMK
    Adodc1.Recordset.MoveNext
Next
    Adodc1.Recordset.MoveFirst

'--------------------------------
'Mengisi Combo Jam dan Menit Sesuai aturan
```

```
For i = 1 To 24
    ComboJam.AddItem i
Next
For i = 0 To 59
    ComboMenit.AddItem i
Next
'----------------------------------------
End Sub
```

Untuk memperindah aplikasi yang dibuat maka dibutuhkan sedikit animasi warna seperti pada uraian sebelumnya. Berikut ini merupakan hasil pembuatan *syntax* tersebut.

```
'Proses Animasi Warna
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
Image3.Visible = False
Image1.Visible = True

End Sub

Private Sub Image1_MouseMove(Button As Integer, Shift As Integer, X As
Single, Y As Single)
Image3.Visible = True
Image1.Visible = False

End Sub
```

Image 2 digunakan sebagai tombol untuk kembali ke menu sebelumnya yaitu menu utama. *Syntax* yang diterapkan pada bagian ini sangatlahh sederhana seperti cuplikan dibawah.

```
'Kembali ke Menu Utama
Private Sub Image2_Click()
Load Form1
Form1.Show
Unload Me
      End Sub
```

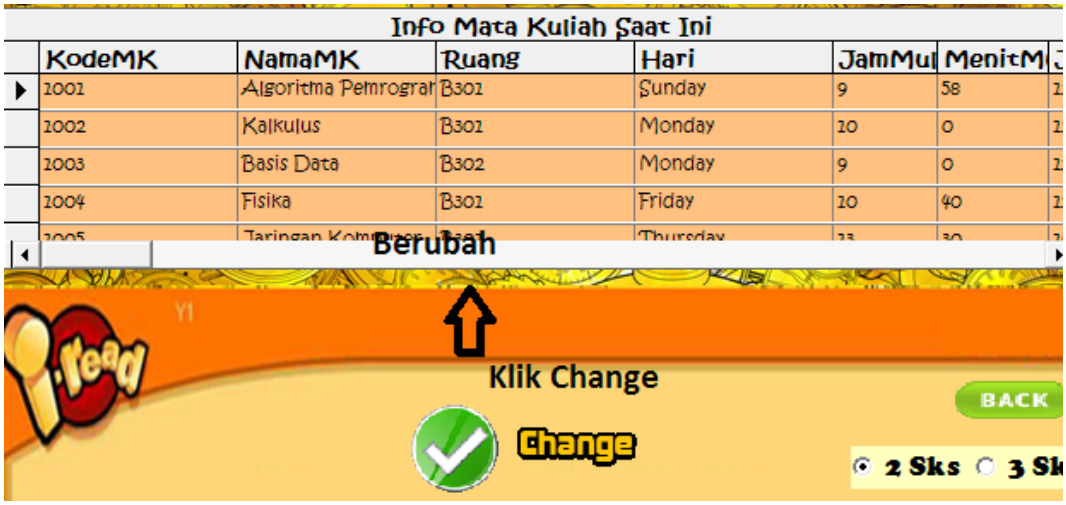
Pada form kedua ini terdapat dua bagian pemrograman yang utama yaitu proses pengisian otomatis saat *combokode* dipilih dan pada image 3 yang digunakan sebagai tombol untuk *update database*.

Gambar 3.24 Pengisian Otomatis

Berikut ini adalah cuplikan program untuk membuat pengisian secara otomatis.

```
'Proses Pengisian otomatis saat memilih Kode MK
Private Sub combokode_Click()
'Poses pengisian object
For i = 0 To Adodc1.Recordset.RecordCount - 1
    If Combokode.Text = Adodc1.Recordset.Fields!KodeMK Then
        TxtNama.Text = Adodc1.Recordset.Fields!NamaMK
        ComboHari.Text = Adodc1.Recordset.Fields!hari
        ComboJam.Text = Adodc1.Recordset.Fields!JamMulai
        ComboMenit.Text = Adodc1.Recordset.Fields!MenitMulai
        ComboRuang.Text = Adodc1.Recordset.Fields!Ruang
    Else
        Adodc1.Recordset.MoveNext
    End If
Next
    Adodc1.Recordset.MoveFirst
'----------------------------------------
'Mengizinkan user untuk mengubah Object lain
TxtNama.Enabled = True
ComboHari.Enabled = True
ComboJam.Enabled = True
ComboMenit.Enabled = True
ComboRuang.Enabled = True
'----------------------------------------
End Sub
```

Proses yang terakhir terletak pada image3 yang dialih fungsikan sebagai tombol *update*. Seluruh pengisian *textbox* dan *combobox* akan diolah dibagian ini dan dimasukkan kedalam *database* untuk disimpan.



Gambar 3.25 Ilustrasi Tombol *Change*

Berikut ini merupakan cuplikan program tersebut.

```
'Image Proses Change atau Update data
Private Sub Image3_Click()
Dim menit As Integer
For i = 0 To Adodc1.Recordset.RecordCount - 1
    'Mengubah isi seluruh atribut pada dengan Acuan Kode MK
    If Combokode.Text = Adodc1.Recordset.Fields!KodeMK Then
        Adodc1.Recordset.Fields!NamaMK = TxtNama.Text
        Adodc1.Recordset.Fields!hari = ComboHari.Text
        Adodc1.Recordset.Fields!JamMulai = ComboJam.Text
        Adodc1.Recordset.Fields!MenitMulai = ComboMenit.Text
        Adodc1.Recordset.Fields!Ruang = ComboRuang.Text
        menit = (ComboJam.Text * 60) + ComboMenit.Text
        'Melihat kondisi Option jika 2sks maka + 100 menit
        If Option1.Value Then
            menit = menit + 100
        ElseIf Option2.Value Then
            menit = menit + 150
        End If
        '-------------------------------------------------
        'Jam dan Menit selesai secara otomatis terhitung
        Adodc1.Recordset.Fields!JamSelesai = menit / 60
        Adodc1.Recordset.Fields!MenitSelesai = menit Mod 60
        '-------------------------------------------------
    Else
        Adodc1.Recordset.MoveNext
    End If
Next
    Adodc1.Recordset.MoveFirst
    '-----------------------------------------------------
End Sub
```

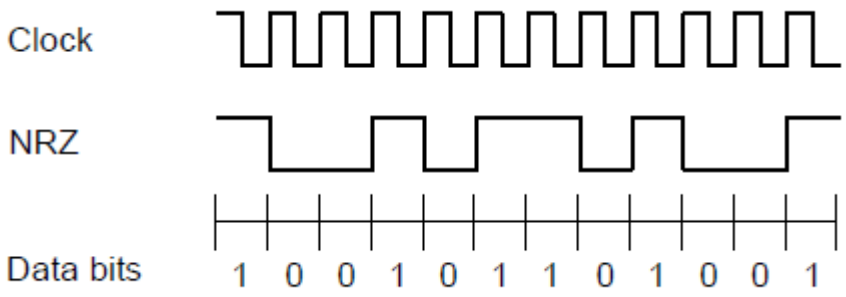
3.5 Perbandingan Berbagai Metode

Metode pengiriman data merupakan metode yang digunakan untuk melakukan pemindahan data dari pengirim ke penerima. Sebuah sinyal digital biasanya berupa gelombang kotak.

Modul wireless mengharuskan pengiriman data dengan menggunakan gelombang kotak dan memiliki keluaran tegangan 0 volt dan 5 volt (hanya mendukung 2 level tegangan). Untuk memenuhi kebutuhan hardware ini dibutuhkan suatu metode pengiriman data yang mendukung. Pada sub bab selanjutnya akan dibahas 3 buah metode pengiriman data yaitu metode NRZ, metode Manchester, dan Metode MLT-3.

3.5.1 Metode NRZ

Menurut George (2010) metode NRZ (Non-Return to Zero) merupakan metode pengiriman data yang paling sederhana karena hanya menggunakan 2 level tegangan dan mempresentasikan 1 sebagai tegangan tertinggi dan 0 sebagai tegangan terrendah.



Gambar 3.26 Metode NRZ

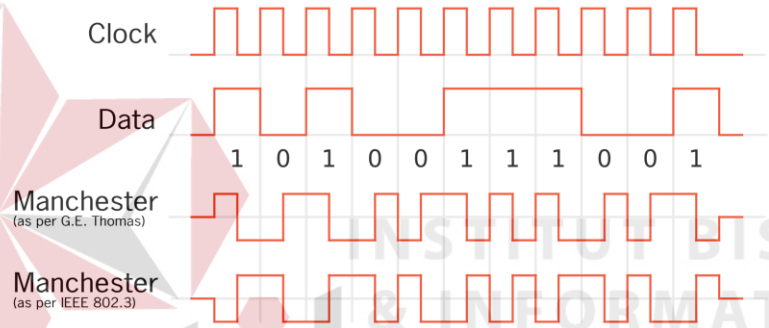
3.5.2 Metode Manchester

Menurut Millis (2009) Setiap transmisi data memiliki sebuah aturan yang harus dilakukan. Aturan pada Manchester *coding* adalah sebagai berikut :

1. Jika data sebenarnya adalah logika 0, maka kode Manchester adalah 0 ke 1.
2. Jika data sebenarnya adalah logika 1, maka kode Manchester adalah 1 ke 0.

Aturan pada Manchester *coding* (Sesuai dengan IEEE) :

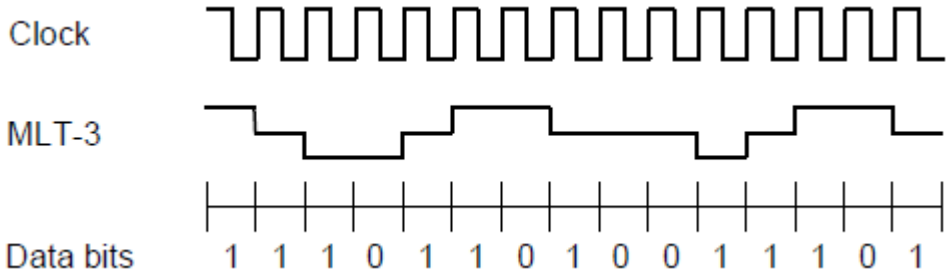
1. Jika data sebenarnya adalah logika 0, maka kode Manchester adalah 1 ke 0.
2. Jika data sebenarnya adalah logika 1, maka kode Manchester adalah 0 ke 1.



Gambar 3.27 Manchester *Code*

3.5.3 Metode MLT-3

Menurut George (2010) MTL (Multi Level Threshold) digunakan untuk mengurangi konten sinyal dengan frekuensi yang tinggi. MLT-3 menggunakan 3 level tegangan yang ditunjukan oleh -1, 0, 1. Sekali process cycles melalui empat nilai -1, 0, +1, 0. Perubahan tingkatan level menunjukan arti data asli adalah 1, sedangkan jika tidak terdapat perubahan tingkatan level maka berarti data asli adalah 0. Gambar 3.28 memperlihatkan contoh pengiriman menggunakan metode MLT-3.

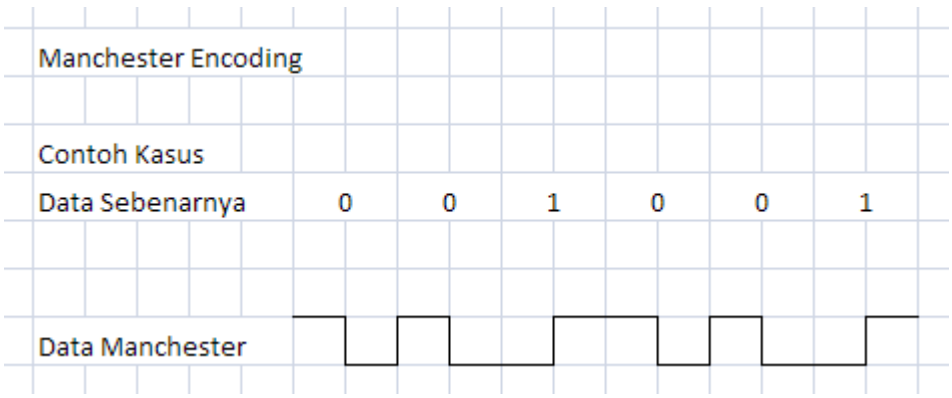


Gambar 3.28 Metode MLT-3

3.6 Pemilihan Metode Manchester

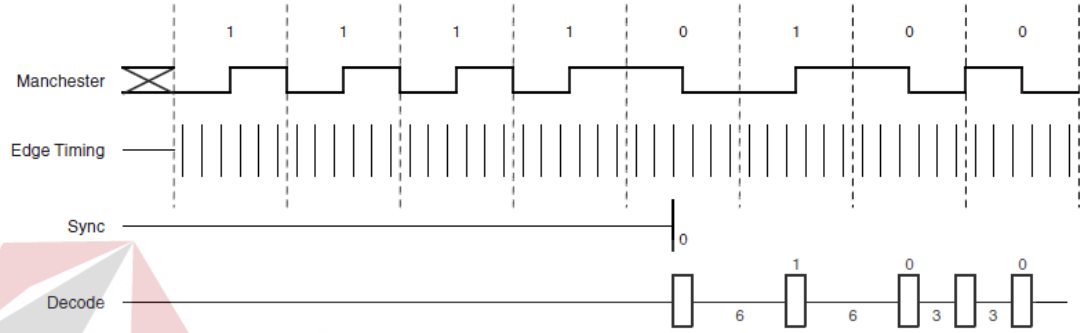
Metode NRZ tidak dapat digunakan pada modul wireless RWS dan TWS 434 karena metode ini tidak membangkitkan gelombang, sedangkan metode MLT-3 tidak dapat digunakan karena metode MLT-3 menggunakan 3 level tegangan, dimana modul tersebut hanya mendukung 2 level tegangan. Oleh karena itu pemilihan metode Manchester digunakan sebagai metode pengiriman data pada aplikasi dikarenakan alasan kebutuhan hardware.

Pada penelitian kali ini digunakanlah aturan Manchester sesuai dengan IEEE yang terlihat pada gambar 3.27 bagian bawah. Proses *encoding* pada Manchester jauh lebih mudah jika dibandingkan dengan proses *decoding*nya.



Gambar 3.29 Ilustrasi Manchester *Encoding*

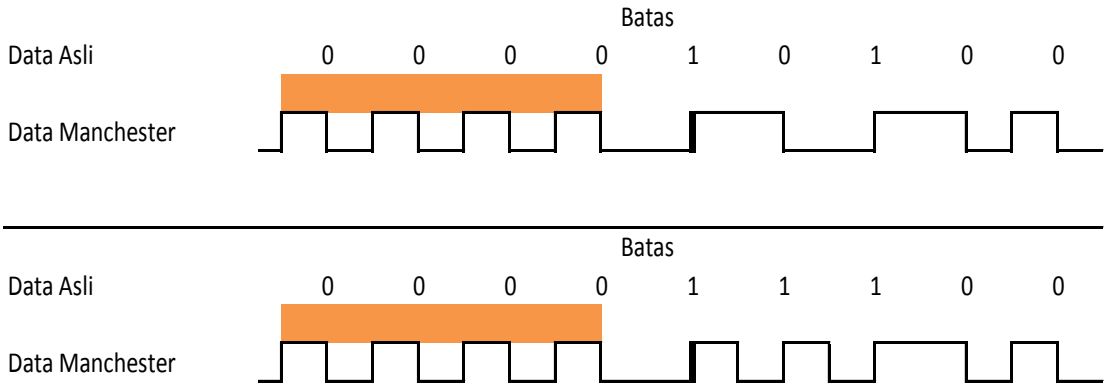
Gambar 3.29 mengilustrasikan gambaran data saat melakukan *encoding* data asli ke data Manchester. Proses *decoding* merupakan proses yang digunakan untuk mengembalikan data Manchester menjadi data asli pada sisi penerima, pada bab landasan teori telah dijelaskan bahwa terdapat 2 macam algoritma yang dapat digunakan yaitu *sampling based* dan *timming based*.



Gambar 3.30 Metode *Sampling based*

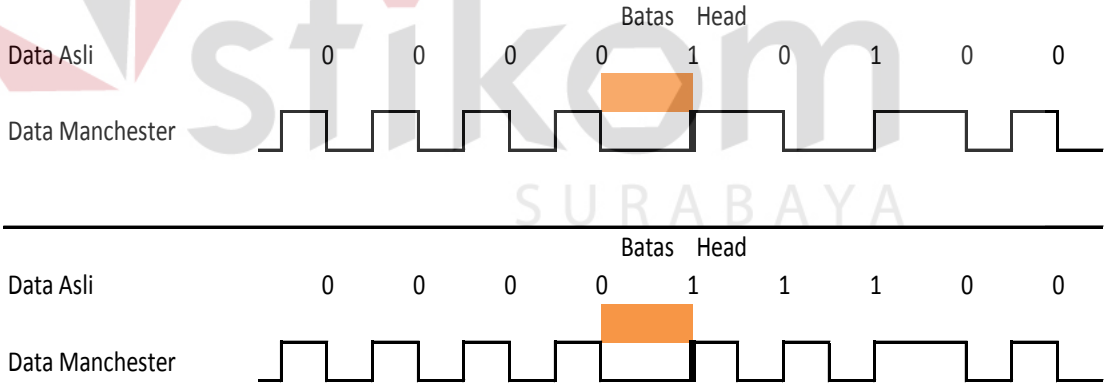
Pada penetilian ini digunakanlah algoritma *sampling based* yaitu dengan melakukan sampel seperti yang terlihat pada gambar 3.30. Secara sederhana proses *decoding* dapat dijabarkan menjadi 3 proses utama yaitu proses sinkronisasi, pendeteksi *header*, dan pembacaan.

Proses sinkronisasi terjadi saat pertama kali program membaca keadaan data. Pada proses ini dilakukan sampel dan penghitungan rata-rata yang digunakan untuk mengetahui berapa banyak hitungan sampel pada kondisi *high* dan *low* (setengah gelombang Manchester). Satu data asli akan menjadi 1 gelombang data Manchester.



Gambar 3.31 Ilustrasi *Decoding* Sinkronisasi

Algoritma yang peneliti gunakan harus memperhatikan perubahan data *high* ke *low* atau sebaliknya. Sehingga pada proses inisialisasi digunakanlah gelombang stabil pada Manchester yang membentuk siklus stabil untuk mengetahui jumlah ketukan setengah gelombang Manchester.

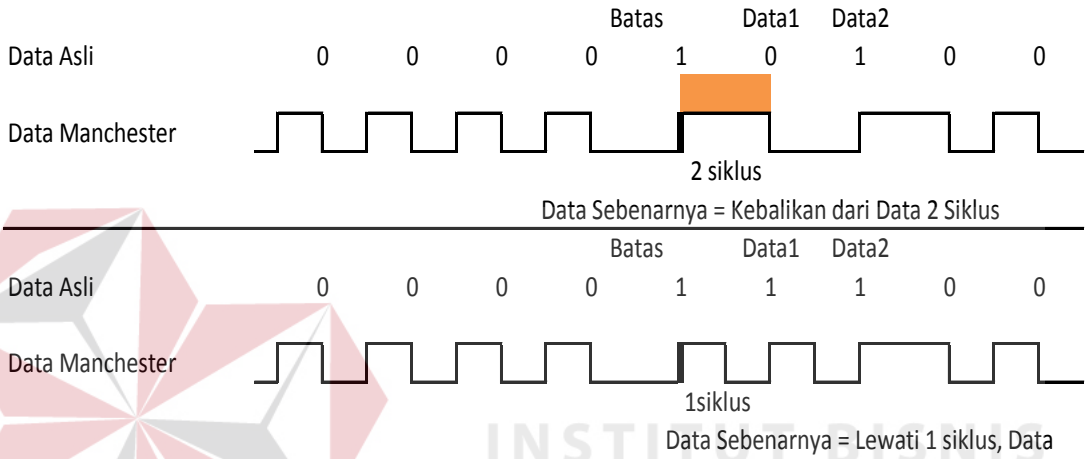


Gambar 3.32 Ilustrasi *Decoding* Mencapai *Header*

Jika rata-rata hitungan telah diketahui maka saat hitungan data > (lebih besar) dari rata-rata maka pembacaan telah mencapai batas data. Perhatikan

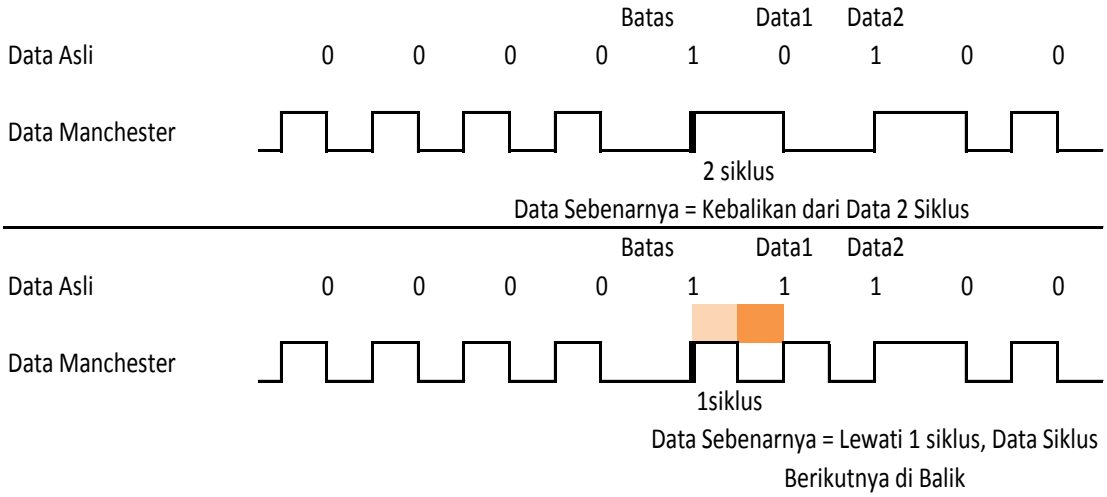
bahwa posisi pointer pembacaan saat ini berada tepat pada garis vertical dengan cetak tebal pada gambar 3.32.

Setelah mendapatkan posisi batas *header* maka proses berikutnya adalah pembacaan data. Pembacaan data diasumsikan terjadi 2 buah data yang berlainan yaitu data setelah *header* bernilai 0 dan data bernilai 1.



Gambar 3.33 Ilustrasi *Decoding* Satu Gelombang

Saat data setelah *header* bernilai nol maka hasil hitungan untuk data berikutnya adalah 1 gelombang (lebih besar dari rata-rata) maka kebalikan data hitungan tersebut adalah data asli. Pada gambar 3.33 terlihat bahwa data pembacaan *high* sehingga data sebenarnya adalah 0.



Gambar 3.34 Ilustrasi *Decoding* Setengah Gelombang

Namun disaat data berikutnya setelah *header* bernilai satu maka akan membentuk siklus setengah gelombang maka data siklus setengah gelombang pertama diabaikan dan kebalikan dari data siklus setengah gelombang berikutnya adalah data asli.

Demikianlah seterusnya hingga banyaknya pembacaan data yang diinginkan terpenuhi. Penerapan metode Manchester pada penelitian kali ini harus menggunakan metode seperti ini karena komunikasi yang digunakan adalah komunikasi bertipe asinkronus.

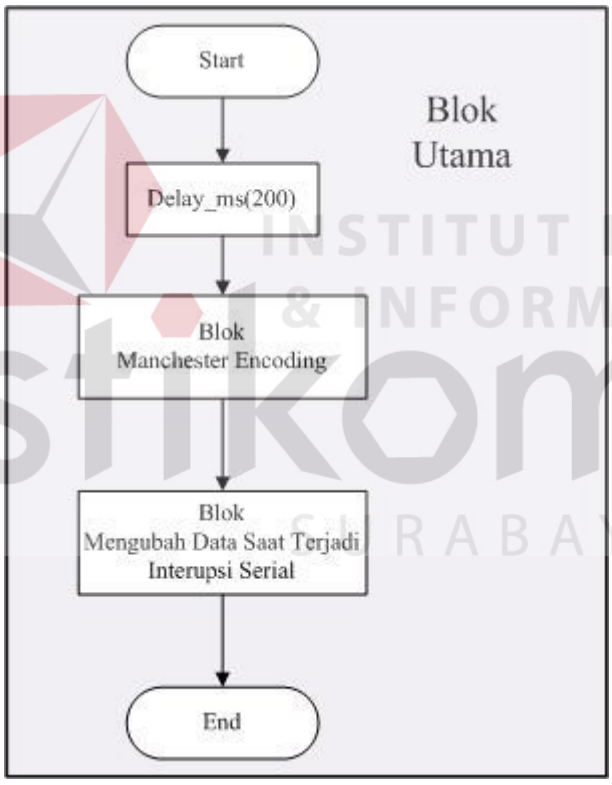
3.7 *Flow Chart* Mikrokontrol

Manchester coding merupakan algoritma pengiriman data yang di tanamkan kedalam mikrokontroler. Algoritma transmisi data ini terbagi menjadi dua bagian utama yaitu Manchester *encoding* (*transmitter*) dan Manchester *decoding* (*receiver*).

Penjelasan mengenai *flowchart* diatas akan dipecah-pecah menjadi beberapa bagian agar mempermudah dalam penjelasan. Sub bab berikutnya akan membahas mengenai kedua *flowchart* besar diatas.

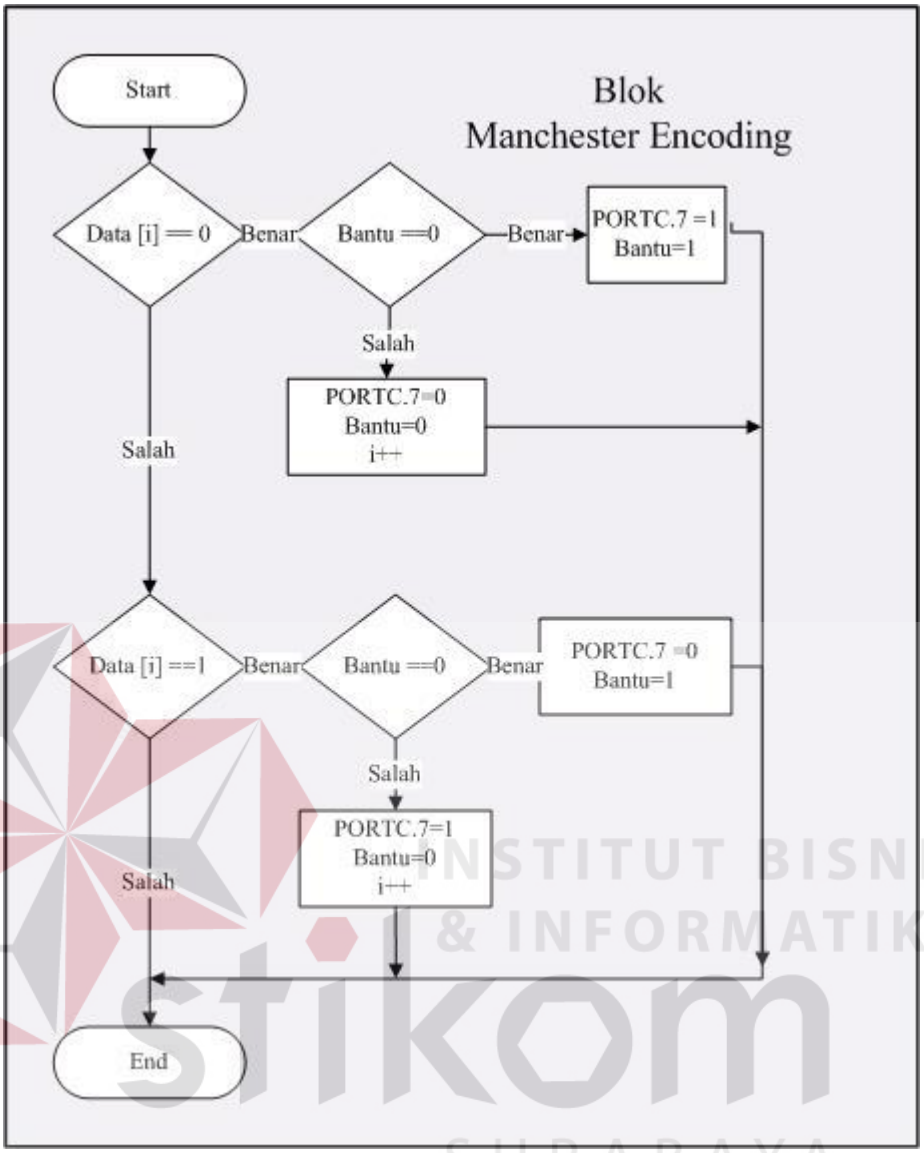
3.7.1 *Flow Chart Transmitter*

Flow chart pada sisi *transmitter* terbagi menjadi 2 bagian utama yaitu *main* program terlihat pada gambar 3.35 dan sub program *interrupt* serial terlihat pada gambar 3.38.



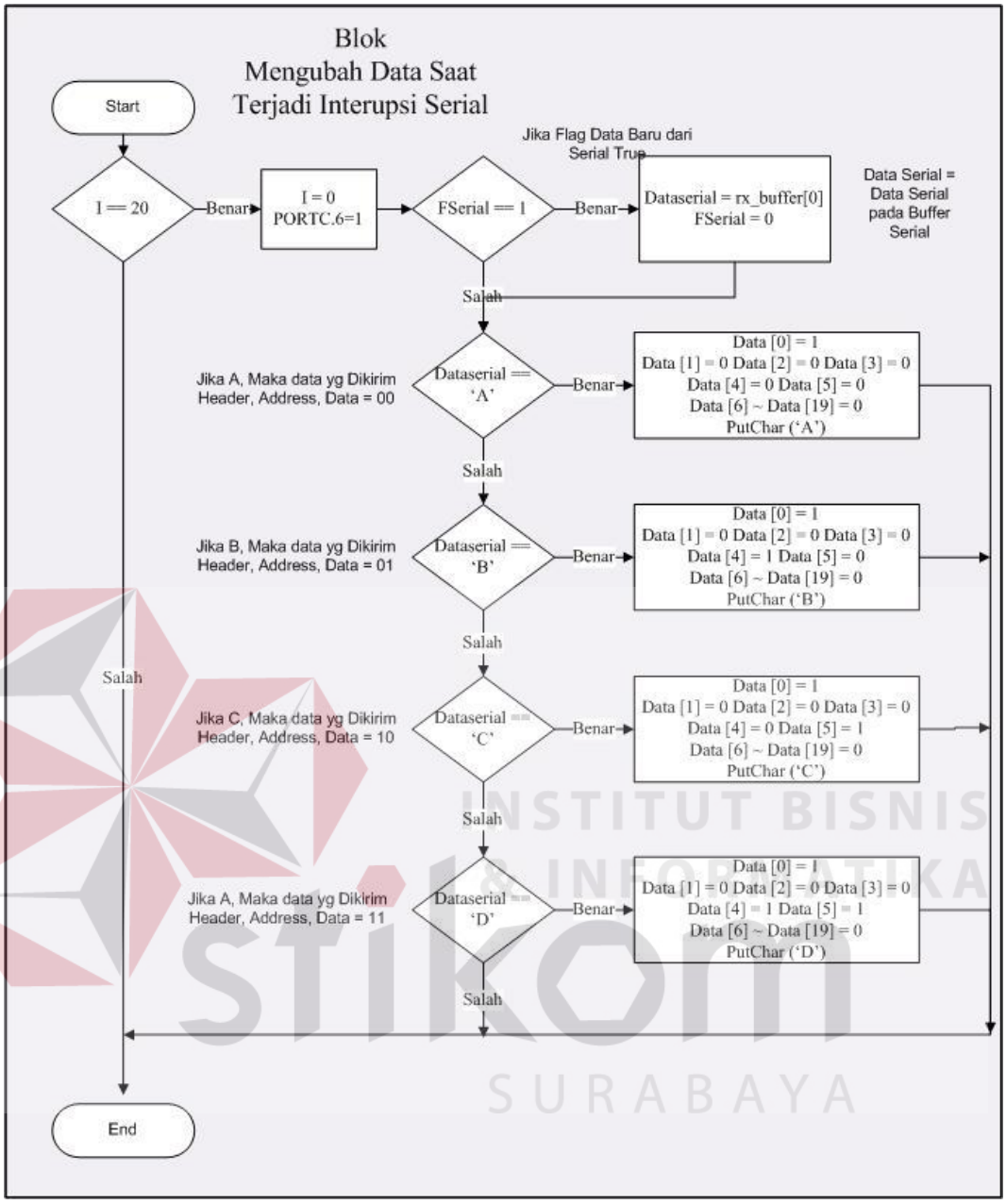
Gambar 3.35 *Flowchart* Blok Utama *Encoding*

Flow chart blok utama memiliki 3 blok yaitu *delay*, Manchester *encoding*, dan pengubah data. *Delay* digunakan untuk mengatur kecepatan pengiriman data yang dilakukan.



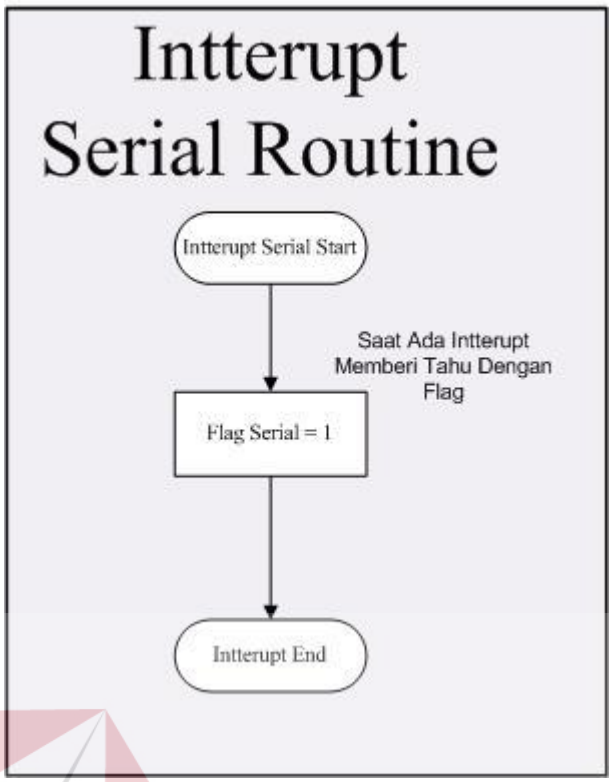
Gambar 3.36 *Flowchart* Blok Manchester *Encoding*

Blok Manchester berisi mengenai algoritma yang digunakan untuk mengubah data digital menjadi data Manchester, yaitu data 0 diubah menjadi *high low* dan data 1 menjadi *low high*. Untuk mengubah menjadi data Manchester, data sebenarnya harus ditransmisikan dua kali sehingga digunakanlah sebuah variabel bantu. PORTC.7 merupakan port yang terhubung dengan pin modul TLP secara langsung, digunakan untuk mengirimkan sinyal ke modul.



Gambar 3.37 Blok Pengubah Data *Encoding*

Blok pengubah data merupakan blok untuk melakukan *refresh* ulang data yang akan dikirimkan.

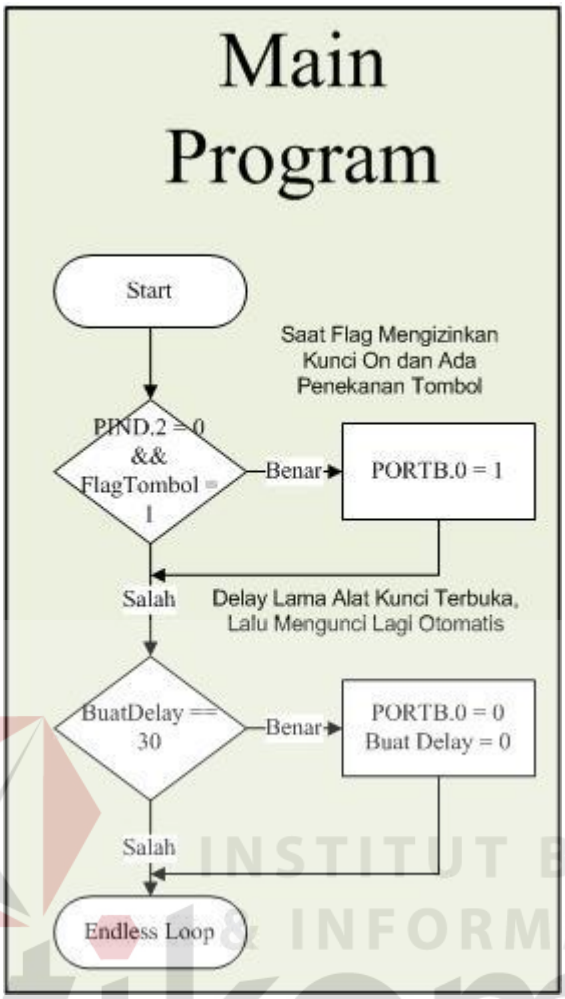


Gambar 3.38 *Flowchart* Blok *Interrupt* Serial Routine *Encoding*

Flow chart blok *interrupt* serial routine digunakan untuk memberikan *flag* manual guna menandakan bahwa terdapat data baru yang masuk.

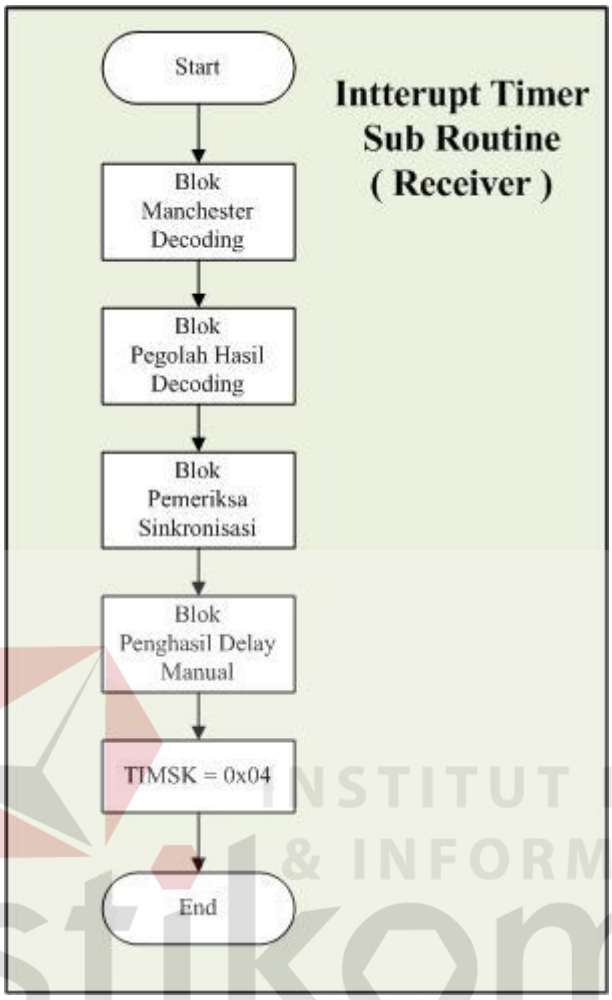
3.7.2 *Flow Chart Receiver*

Flow chart pada sisi *receiver* terbagi menjadi 2 bagian utama yaitu *main* program terlihat pada gambar 3.39 dan sub program *interrupt timer* terlihat pada gambar 3.40.



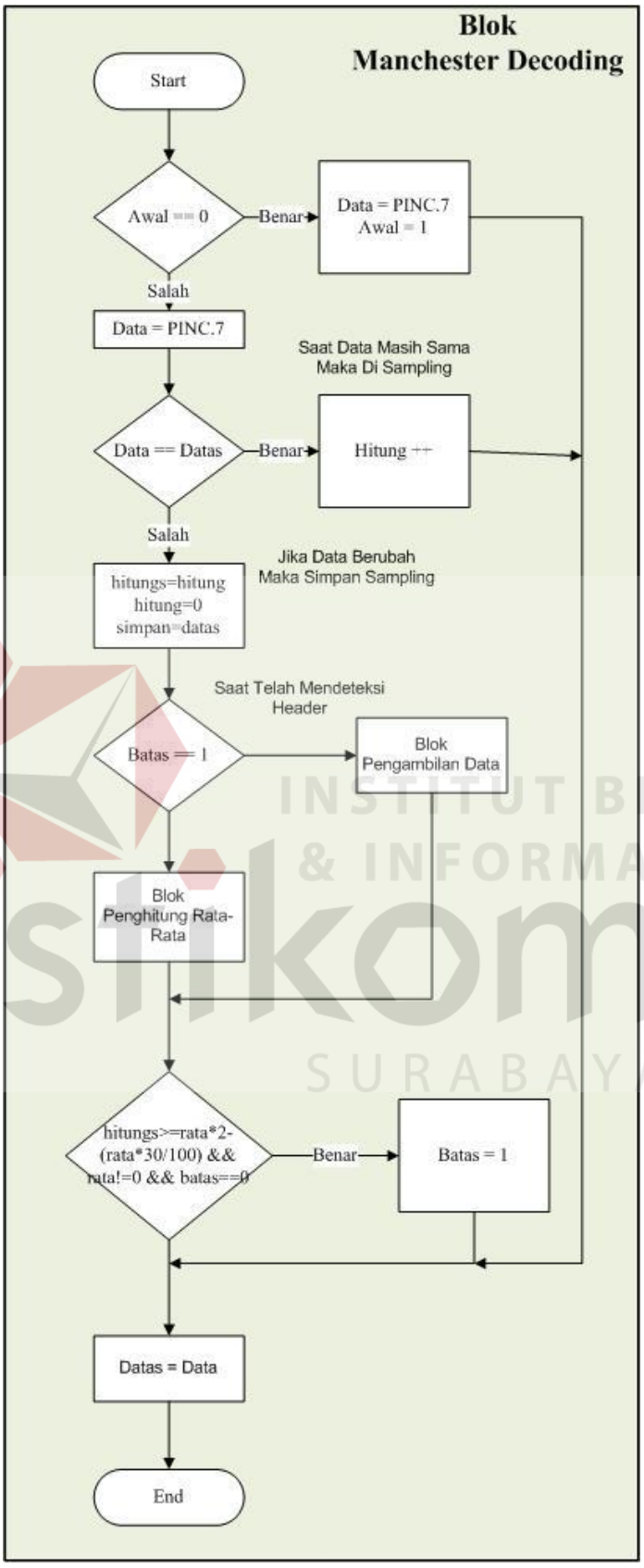
Gambar 3.39*Flowchart* Blok *Main* Program *Decoding*

Secara umum fungsi dari *main* program ini sebagai pengkontrol alat penguncian pintu agar terbuka dan tertutup secara otomatis. Dibutuhkan penekanan tombol (PIND.2) dan status penguncian (*flag* tombol) telah menyatakan terbuka maka alat pengunci akan menyala.



Gambar 3.40 *Flowchart* Blok *Interrupt Timer* Sub Routine *Decoding*

Bagian kedua dari *flowchart receiver* terdapat pada *interrupt timer* terlihat pada gambar 3.40. Blok ini merupakan blok utama dari sistem yang digunakan sebagai penterjemah data Manchester. Secara garis besar blok ini terbagi lagi menjadi 4 blok yaitu blok manchster *decoding*, blok pengolah hasil, blok pemeriksa sinkronisasi, dan blok penghasil *delay* manual.



Gambar 3.41 *Flowchart* Blok Manchester *Decoding*

Gambar 3.41 merupakan gambaran *flowchart* blok untuk melakukan *decoding* pada data Manchester. Proses inisialisasi dilakukan pada awal algoritma untuk melakukan penyimpanan data Manchester yang disampel pertama kali. Hal tersebut bermaksud agar terdapat data awal pada buffer sehingga dapat digunakan pada proses pembandingan. Proses pembandingan berfungsi untuk mendeteksi adanya perubahan data dari 0 ke 1 atau dari 1 ke 0.

Selama data awal masih sama dengan hasil sampel berikutnya maka berarti data belum terjadi perubahan. Jika sampel yang dibaca belum terjadi perubahan maka dilakukan proses penghitungan untuk mengetahui berapa kali sampel dilakukan pada setengah pulsa gelombang.

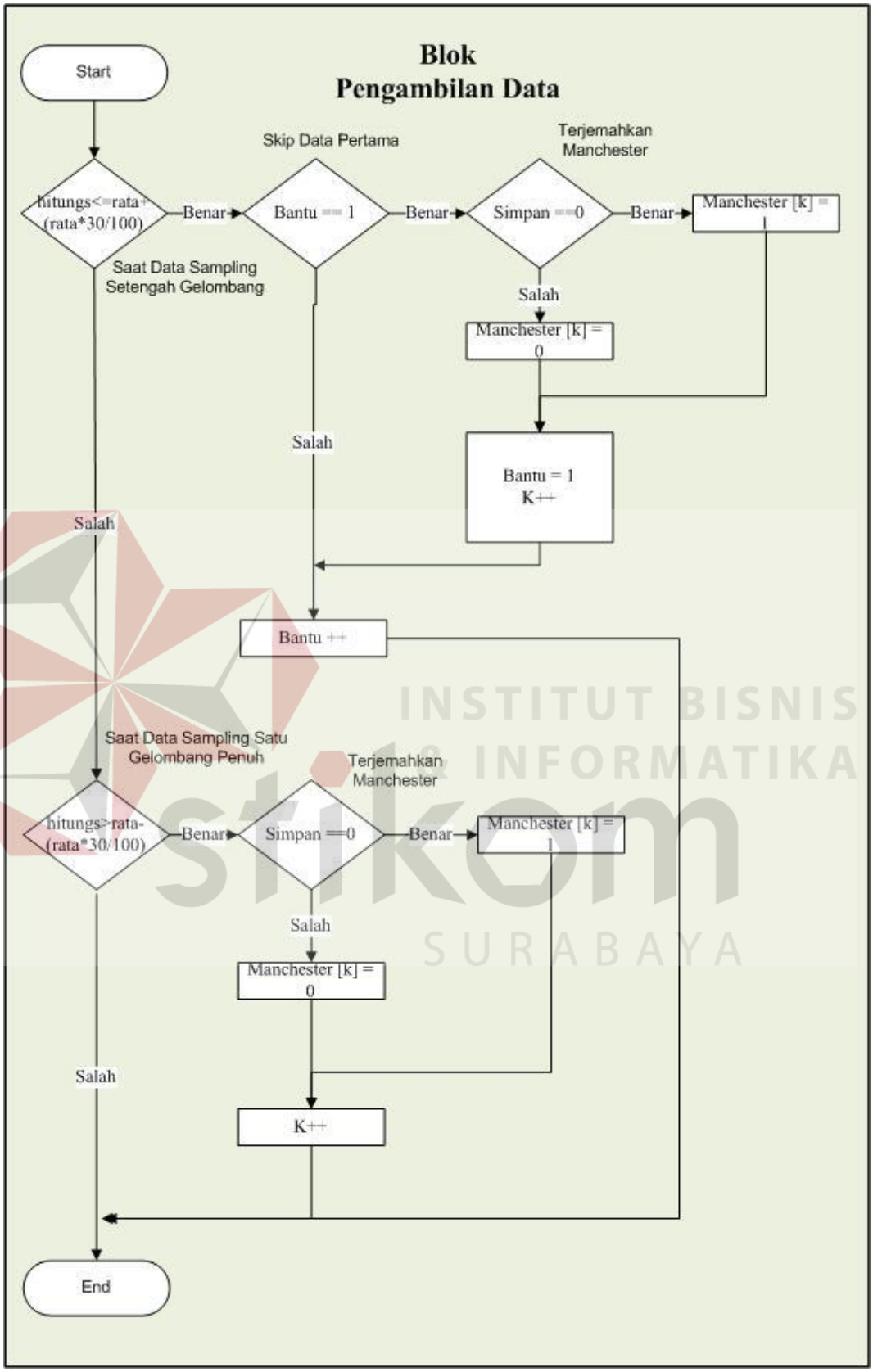
Proses yang utama adalah saat terjadi perubahan data Manchester yang terbaca, karena disaat inilah penghitungan, pengecekan *header* data, dan lainnya terjadi. Oleh sebab itu, sebelum data lama dibuang maka data tersebut seluruhnya disimpan dalam variabel. Data yang disimpan adalah data yang terbaca sebelumnya (data Manchester 0 atau 1) dan data hasil hitungan saat sampel data Manchester. Sebagai contoh data Manchester sebelumnya adalah 0 dan terjadi selama 10 kali ketukan atau pembacaan data sampel. Sehingga berarti bahwa lama data 0 adalah 10 kali ketukan.

Proses berikutnya adalah pendeteksi batas serta proses *decoding* berikutnya jika telah mencapai batas. Proses utama berikutnya terjadi disaat batas telah tercapai, oleh karena itu urutan untuk proses pendeteksi batas dan proses pembacaan berikutnya diletakkan pada urutan yang terbalik.

Saat pendeteksi batas telah memberikan *flag* maka akan dilakukan proses pembacaan data, namun saat batas belum ditetapkan maka proses yang dilakukan

adalah proses penghitungan rata-rata guna mengetahui banyaknya jumlah ketukan pada setengah gelombang. Hasil rata-rata inilah yang digunakan sebagai acuan untuk pembacaan data Manchester. Blok pencari rata-rata dan pengambilan data akan dipaparkan pada *flowchart* berikutnya.

Selama data belum mencapai batas maka data sebenarnya yang dikirimkan adalah 0 sehingga jika diubah menjadi kode Manchester menjadi *high low* secara terus menerus. Oleh sebab itu *header* hanya akan terjadi disaat hasil ketukan melebihi rata-rata dari perhitungan data sinkronisasi. Sehingga dapat dirumuskan menjadi "Hitungs >= rata*2 – (rata*30/100) && rata != 0 && batas ==0" digunakan persentase error sebesar 30% yang mana persentase error maximal tidak boleh melebihi 50%. Penentuan batas diamankan lagi dengan syarat bahwa *flag* batas belum dinyalakan dan rata-rata memiliki nilai.



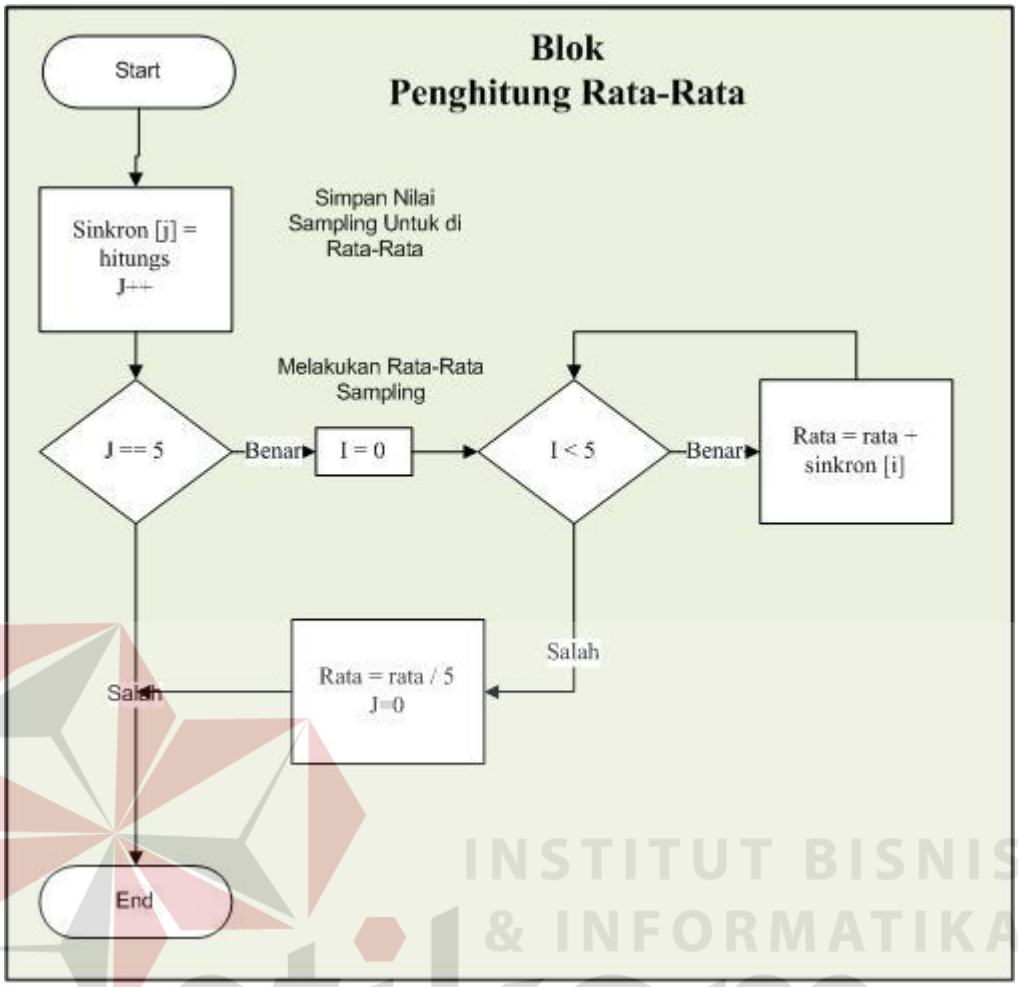
Gambar 3.42 *Flowchart* Blok Pengambilan Data *Decoding*

Pada gambar 3.42 merupakan gambaran blok pengambilan data Manchester yang dilakukan saat *flag* batas telah dinyalakan. Untuk pembacaan hasil *decoding* telah dipaparkan secara mendetail pada sub bab metode Manchester. Secara garis besar algoritma yang dilakukan adalah sebagai berikut.

Saat pembacaan setelah *header* adalah setengah gelombang (dimana besar hitungan ketukannya kurang atau sama dengan nilai rata-rata) maka pembacaan data setengah gelombang pertama dilewatkan terlebih dahulu dengan memanfaatkan variabel bantu. Maka kebalikan data pembacaan berikutnya merupakan data asli (jika pembacaan berikutnya 0 maka data asli adalah 1 dan sebaliknya).

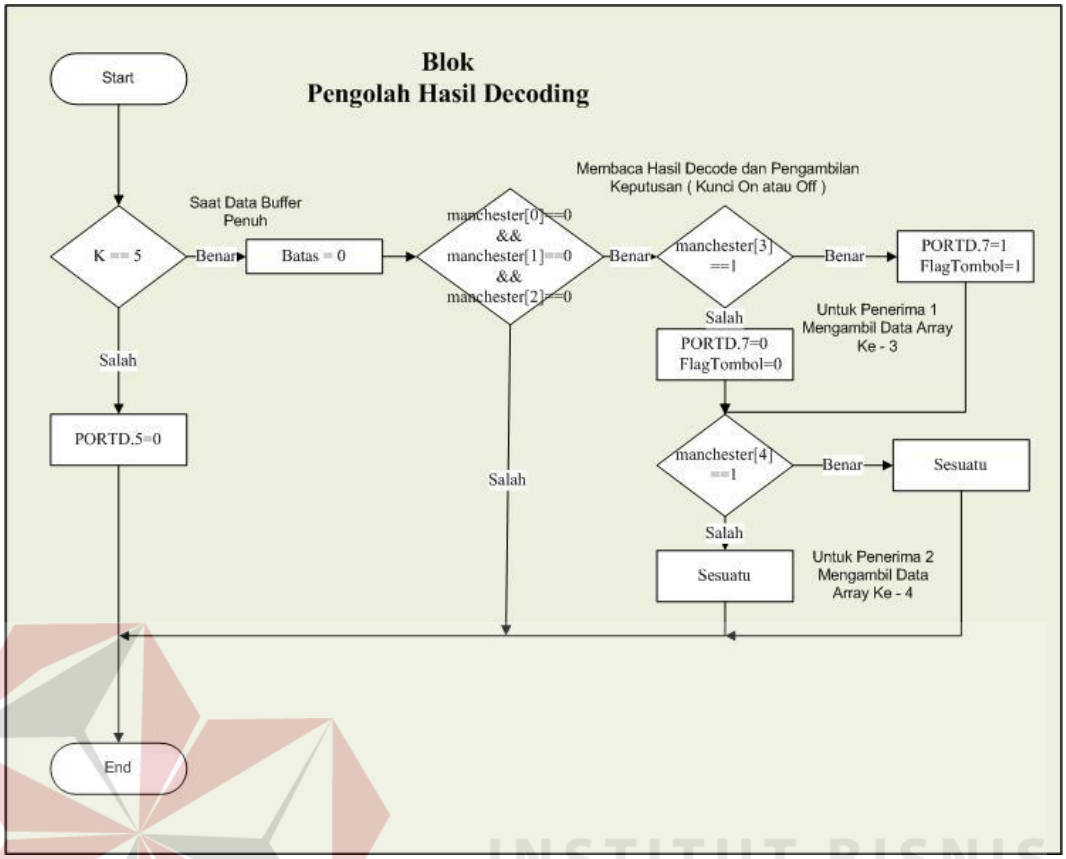
Saat pembacaan setelah *header* adalah satu gelombang (dimana besar hitungan ketukannya >= rata-rata) maka kebalikan data pembacaan berikutnya merupakan data asli (jika pembacaan berikutnya 0 maka data asli adalah 1 dan sebaliknya).

Seluruh hasil pembacaan data Manchester yang telah diubah menjadi data asli disimpan kedalam variabel *array* bernama Manchester. Banyaknya pembacaan yang dilakukan dapat diatur dengan menambah jumlah *array* pada proses ini. Ini merupakan salah satu kelebihan menggunakan metode Manchester yaitu jumlah data yang dapat diatur sesuai kebutuhan.



Gambar 3.43 *Flowchart* Blok Penghitung Rata-Rata *Decoding*

Blok penghitung rata-rata dapat dilihat pada gambar 3.43. Algoritma yang digunakan untuk mencari rata-rata sangatlah umum digunakan. Selama data rata-rata belum mencapai batas (dalam hal ini 5) maka data rata-rata akan disimpan ke dalam variabel *array*. Saat data telah siap untuk dioperasikan maka dilakukan perulangan untuk menjumlahkan data. Jika data telah dijumlahkan seluruhnya maka data tersebut dibagi sebanyak jumlah data (dalam hal ini 5). Hasil dari proses penjumlahan dan pembagian tersebut merupakan hasil rata-rata data.



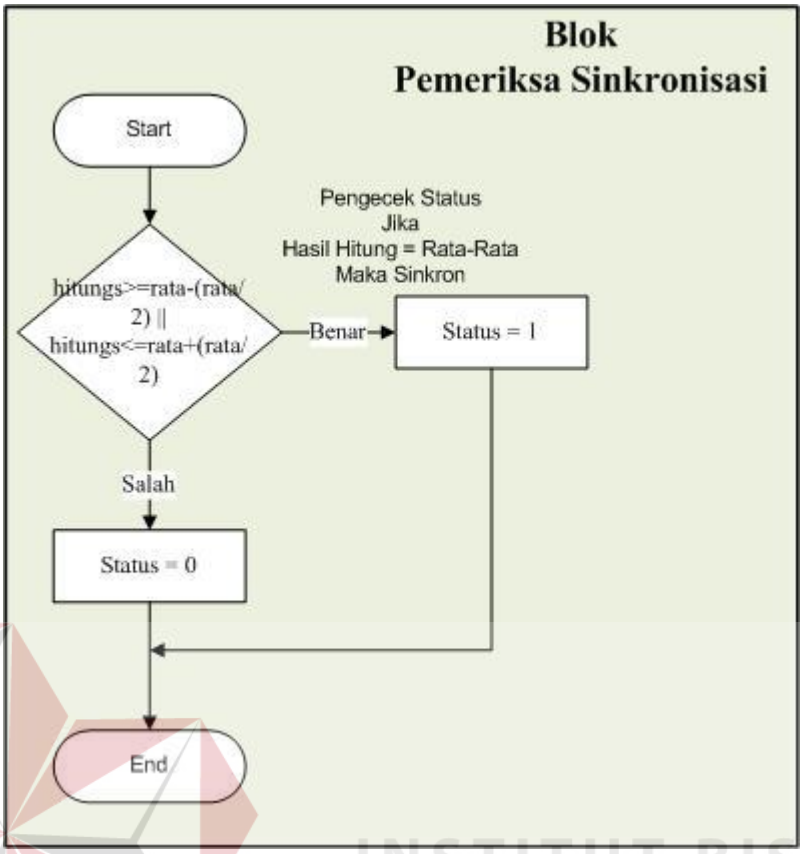
Gambar 3.44 *Flowchart* Blok Pengolah Hasil *Decoding*

Pada Blok sebelumnya telah dijelaskan mengenai pengambilan data Manchester menjadi data asli yang tersimpan pada variabel *array* bernama Manchester. Gambar 3.44 merupakan proses penggunaan data asli yang telah didapatkan pada blok sebelumnya. Selama jumlah K != 5 maka data belum dapat diproses, dimana K merupakan variabel yang menentukan jumlah pembacaan data asli (banyaknya data yang ingin dibaca, salah satu kelebihan Manchester).

Saat buffer variabel *array* telah mencapai batas pembacaan maka *flag* batas kembali dimatikan dan dilakukan proses penggunaan data asli. Dalam kasus ini terdapat 3 *bit* data yang digunakan sebagai alamat sistem dan 2 *bit* data yang digunakan untuk pengaturan. Jika data alamat adalah 000 (data alamat dapat dikatakan sebagai password) maka dilakukan proses selanjutnya.

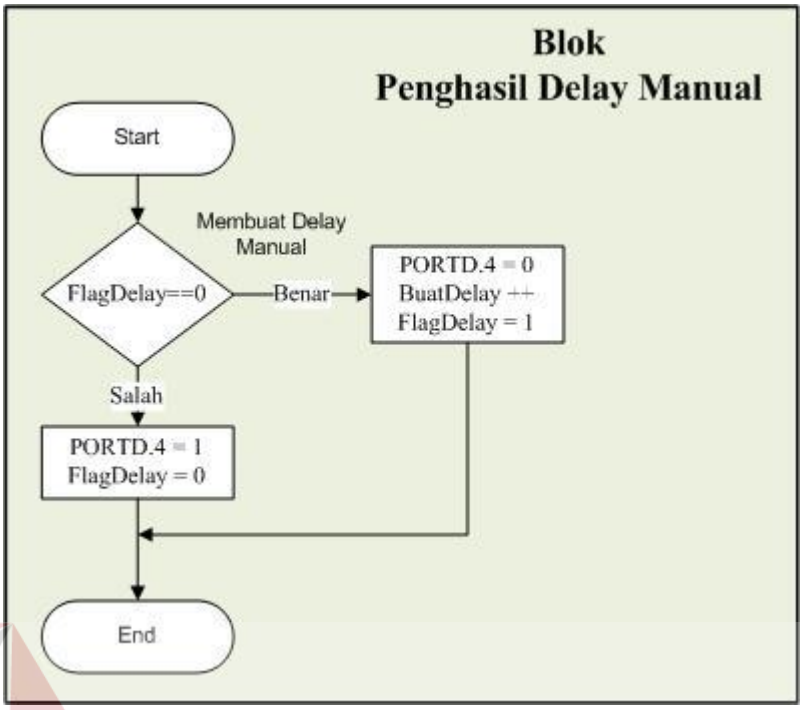
Sistem penguncian pintu menggunakan sebuah *transmitter* dan dua buah *receiver* sebagai objek penelitian, oleh sebab itu untuk mengkontrol *Electric Door Lock* hanya dibutuhkan 1 bit data saja. Karena *receiver* berjumlah dua buah maka digunakanlah bit data sebanyak dua buah yaitu bit *array* index ke-3 dan ke-4. Untuk *receiver* pertama didownload *syntax* (PORTD.7 dan *Flag*Tombol) pada *array* index ke-3 sedangkan *array* index ke-4 dibiarkan kosong ("sesuatu" pada blok merupakan opsi saat mendownload *receiver* kesatu atau kedua), sehingga saat mendownload *receiver* kedua index ke-3 dikosongkan dan index ke-4 ("sesuatu") diubah menjadi *syntax* untuk mengakses PORTD.7.

Syntax PORTD.7 digunakan sebagai indikator untuk mengetahui bahwa *Electric Door Lock* diperintahkan untuk membuka atau menutup, sedangkan *flag* tombol berfungsi untuk memberitahukan program utama bahwa program mengizinkan pintu terbuka atau tertutup yang digunakan untuk pengambilan keputusan pada program utama (telah dijelaskan pada blok program utama).



Gambar 3.45 *Flowchart* Blok Pemeriksa Sinkronisasi *Decoding*

Secara garis besar penggunaan blok algoritma pada gambar 3.45 ini tidaklah terlalu bermanfaat karena blok ini hanya digunakan untuk memberikan indikator bahwa data sampel sama dengan rata-rata yang dilakukan. Jika hasil hitungan ketukan sama maka dapat dikatakan bahwa pengiriman data telah berlangsung dengan baik (tidak ada halangan diudara ataupun hal lain yang menghambat pengiriman data).



Gambar 3.46 *Flowchart* Blok Penghasil *Delay* Manual *Decoding*

Disaat ber*main* dengan interupsi, terutama saat menggunakan interupsi *timer* maka penggunaan fasilitas *delay* yang telah disediakan pada CVAVR akan mengalami sedikit masalah, oleh sebab itu pada penelitian kali ini digunakan *syntax* sederhana untuk menghasilkan *delay* yang tidak mengganggu interupsi yang digunakan. PORTD.4 digunakan sebagai indikator guna melihat kecepatan sampel data yang dilakukan oleh program, sehingga untuk membuat indikator berkedip digunakanlah *flag delay*.

Variabel buat *delay* digunakan untuk membuat *delay*, sehingga untuk memperoleh waktu *delay* yang lama maka buat *delay* hanya akan bertambah saat telah terjadi dua ketukan (jika ketukan terjadi 4 kali maka buat *delay* bernilai 2). Hal tersebut digunakan untuk memperlama *delay*. Variabel buat *delay* ini juga digunakan pada sub pembahasan pada blok program utama.