LAPORAN PRATIKUM KOMUNIKASI DATA

**PENGKODEAN DATA DIGITAL**

**(NRZ-L, NRZ-I, RZ, MANCHESTER, D-MANCHESTER,AMI)**

**Disusun untuk memenuhi**

**Tugas Mata Kuliah Pratikum Komunikasi Data Semester 2**

**Dosen Pembimbing :**

**Lis Diana Mustafa, S.T., M.T.**



Disusun oleh :

Prastyan Arga Saputra

2231130081

1C TT

# PROGRAM STUDI D-III TEKNIK TELEKOMUNIKASI JURUSAN TEKNIK ELEKTRO

**POLITEKNIK NEGERI MALANG 2023**

* 1. **Tujuan Umum**
     + Membuat program untuk mensimulasikan karakteristik pengkodean data digital (NRZ- L, NRZ-I, RZ, MANCHESTER, DMANCHESTER).
     + Membuat flowchart teknik pengkodean data digital.
  2. **Tujuan Khusus**
     + Memahami macam-macam pengkodean data digital (NRZ-L, NRZ-I, MANCHESTER, D-MANCHESTER, RZ)
     + Memahami teknik pengkodean data digital
     + Mampu melakukan simulasi pengodean data digital
  3. **Alat dan Bahan**

1. Laptop / computer
2. Software Delphi 7
   1. **Teori Penunjang**

**A. Pengkodean sinyal digital**

Data digital merupakan data yang memiliki deretan nilai yang berbeda dan memiliki ciri-ciri tersendiri. Contoh data digital adalah teks, deretan bilangan, dan karakter-karakter yang lain. Data digital dalam bentuk karakter yang dapat dipahami

manusia tidak dapat langsung ditransmisikan dalam system komunikasi, data harus terlebih dahulu diubah ke dalam bentuk biner.Jadi, data digital ditransmisikan dalam bentuk deretan biner.Sedangkan sinyal digital merupakan sinyal untuk menampilkan data digital. Deretan pulsa yang berbeda dan tidak terjadi secara terus-menerus merupakan contoh sinyal digital.

Pengkodean data digital ditujukan untuk membangun sinyal digital, yang meliputi: unipolar, polar dan bipolar. Unipolar merupakan pengkodean paling sederhana dimana data digital yang telah berbentuk biner di diubah ke bentuk pulsa, biner ‘1’ diwujudkan dalam suatu tegangan positif dan biner ‘0’ sebagai tanpa tegangan atau nol volt. Gambar 2.1 menampilkan contohsinyal digital sederhana dari suatu data 101100110B.

Amplitudo

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

*t*

Unipolar

Gambar 2.1 Sinyal digital unipolar dari data 101100110B

Pengkodean polar bercirikan menggunakan dua level tegangan positif dan negative sehingga dapat mengeliminasi timbulnya komponen DC. Pengkodean polar terdiriatas NRZ (non return to zero), RZ (return to zero) dan biphase.

1. **NRZ (*Non-Return To Zero*)**

Format yang paling mudah dalam mentransmisikan sinyal digital adalah dengan menggunakan dua tingkat tegangan yang berlainan untuk dua jenis digit biner. Kode –kode biner dikonversikan ke level tegangan tertentu sesuai dengan nilainya. Tingkat tegangan tetap konstan sepanjang interval bit yang ditransmisikan. Format pengkodean ini dibagi menjadi dua tipe, yaitu:

* 1. *Non-Return to Zero level* (NRZ-L), tegangan negatif dipakai untuk mewakili nilai biner ‘1’ dan tegangan positif dipakai untuk mewakili nilai biner lainnya.
  2. *Non-Return to Zero Inverted* (NRZ-I), suatu transisi (rendah ke tinggi atau tinggi ke rendah) dilakukan pada awal suatu bit apabila ditemukan biner ‘1’ dan tidak ada transisi apabila ditemukan biner ‘0’. NRZ-I merupakan salah satu contoh dari *differensial encoding* (penyandian encoding).

Gambar 2.2 menampilkan perbedaan kedua tipe pengkodean tersebut.

**NRZ-L**

**0**

**1**

**0**

**0**

**1**

**1**

**0**

**0**

**1**

**NRZ-I**

Gambar 2.2 Perbedaan NRZ-L dan NRZ-I

1. **Return To Zero (Multilevel Binary)**

Format pengkodean selalu menuju ke level nol pada setengah periodenya. Biner ‘0’ diwakili oleh perubahan level dari negatif ke nol sedangkan biner ‘1’ diwakili oleh perubahan dari positif ke nol. Gambar 2.3 menampilkan contoh sinyal hasil pengkodean return to zero (RZ).

**0 1 0 0 1 1 0 0 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  |  |  |  |  |  |  |  |
|  |  |  | |  |  |  |  |  | |  | |  |  |  |  |  | |
|  | |  | |  | |  | |  | |

**RZ**

Gambar 2.3Sinyal Return To Zero

1. **Biphase**

Biphase merupakan format pengkodean yang dikembangkan untuk mengatasi keterbatasan kode NRZ. Dikenal dua teknik pengkodean biphase, yaitu *Manchester* dan *DifferensialManchester*.Pada pengkodean *Manchester,*ditandai dengan terjadinya suatu transisi pada setengah periode bit: transisi rendah ketinggi mewakili bit ‘1’ dan tinggi kerendah mewakili bit ‘0’. Sedangkan *differensial Manchester* adalah suatu kode dimana bit ‘0’ diwakili oleh adanya transisi diawal periode dan bit ‘1’ diwakili oleh tidak adanya transisi di awal periode suatu bit. Gambar 2.4 menampilkan perbedaan kedua tipe pengkodean tersebut.

**0 1 0 0 1 1 0 0 1**

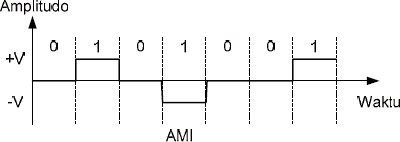
**Manchester**

**D-Manchester**

Gambar 2.4 Perbedaan Manchester dan D-Manchester

1. **AMI**

Pada pengkodean AMI, elemen data dengan bit 1 direpresentasikan oleh sinyal yang beriversi bolak balik dari tegangan positif ke tegangan negatifatau sebaliknya dari tegangan negatif ke tegangan positif.

Sedangkan elemen data dengan bit 0 direpresentasikan oleh tegangan 0 volt.

1. **B8ZS dan HDB3**

Kedua kode ini didasarkan pada pengkodean AMI (Alternative Mark Inversion) dan cocok untuk transmisi dengan kecepatan data tinggi.Ada dua teknik yang umum digunakan dalam layanan transmisi jarak jauh dan keduanya diilustrasikan pada Gambar 8.Pengkodean B8ZS dikenal sebagai pengkodean bipolar dengan 8 nol tertukar sedangkan skema pengkodean didasarkan pada bipolar-AMI. Kelemahan pengkodean ini adalah panjang string (deretan) nol dapat menyebabkan hilangnya sinkronisasi saat transmisi.

Kedua pengkodean ini dibangun untuk menghilangkan deretan bit ‘0’ yang mungkin muncul dalam suatu pengiriman data. B8ZS (*Bipolar* 8-*zero subtitution*) digunakan untuk menggantikan deretan bit ‘0’ lebih dari 8 buah. Aturan pengkodean B8ZS diperlihatkan pada Gambar 2.5, kedelapan bit nol digantikan dengan bit-bit tertentu yang polaritasnya tergantung dari polaritas bit sebelumnya. Bit-bit pengganti bit nol tersebut disebut sebagai bit violasi (*violation code*)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Berubah menjadi

Polaritas bit sebelumnya

violasi

Polaritas bit sebelumnya

violasi

Berubah menjadi

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | 0 | 0 | 0 | + | - | 0 | - | + |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| - | 0 | 0 | 0 | - | + | 0 | + | - |

violasi

violasi

Gambar 2.5 Aturan pengkodean B8ZS

HDB3 (*High-Density Bipolar-3 Zero*) memiliki prinsip yang sama dengan B8ZS, hanya pada HDB3 digunakan untuk menggantikan deretan bit ‘0’ sebanyak 4 buah.

Gambar 2.6 menampilkan aturan penggantian bila ditemukan deretan bit ‘0’ lebih dari 4 buah. Contoh hasil pengkodean menggunakan B8ZS dan HDB3 ditampilkan pada Gambar 2.7.

Polaritas bit sebelumnya

Polaritas bit sebelumnya

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | 0 | 0 | 0 | + |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 0 | 0 | 0 | - |

Jika banyaknya bit ‘1’ sampai bit substitusi adalah ganjil

violasi

violasi

Polaritas bit sebelumnya

Polaritas bit sebelumnya

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | 0 | 0 | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| + | - | 0 | 0 | + |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | + | 0 | 0 | + |

Jika banyaknya bit ‘1’ sampai bit substitusi adalah genap

violasi

violasi

Gambar 2.6Aturan pengkodean HDB3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1 1 0** | **0** | **0** | **0** | **0** | **0** | **0** | **0 1 1 0 0 0 0 0 1** | **0** |  |
|  |  |  |  |  |  |  |  |  | **AMI** |
| **0** | **0** | **0** | **V** | **B** | **0** | **V** | **B** |  |  |
|  |  |  |  |  |  |  |  |  | **B8ZS** |
| **0** | **0** | **0** | **V** | **B** | **0** | **0** | **V B 0 0 V** |  |  |

**HDB3**



Gambar2.7 Contoh pengkodean untuk B8ZS dan HDB3

1. **Teknik Penggambaran Sinyal di Delphi**
   * Menggambar Sinyal RZ

y

*a*

*b*

x

*c*

**1**

**0**

**1**

0,0

Gambar 5. Penggambaran sinyal RZ pada umumnya

Untuk menggambarkan sinyal RZ, koordinat 0,0 terletak pada perpotongan antara sumbu x dan sumbu y. Langkah pertama untuk menggambar adalah melalui koordinat

*a1*

*b1 b2*

0,0,sinyal RZ memiliki 2 kondisi sinyal yang ditandai dengan adanya bit ‘1’ dan bit ‘0’. Dimana untuk bit ‘1’ di wakili oleh gambar sebagai berikut :

Gambar 6. Kondisi sinyal RZ pada posisi bit ‘1’

Pada gambar diatas dapat dilihat bahwa kondisi bit ‘1’ diwakili oleh 4 garis dimana terdapat 2 garis yang mempunyai panjang yang sama (*a1*=*a2*, *b1*=*b2*). Sedangkan untuk bit ‘0’ diwakili oleh gambar sebagai berikut :

*b1*

*a2*

*b2*

*a1*

Gambar 7. Kondisi sinyal RZ pada posisi bit ‘0’

Sama halnya untuk menggambarkan sinyal RZ bit ‘1’, bit ‘0’ juga memiliki 4 garis dimana terdapat 2 garis yang mempunyai panjang yang sama.

* Penggambaran sinyal RZ di Komputer

y 0,0

*a*

*b*

x

*c*

**1**

**0**

**1**

*t*

Gambar 8. Penggambaran sinyal RZ di komputer

Untuk menggambarkan sinyal RZ di komputer sangat berbeda dengan menggambarkan sinyal RZ pada umumnya. Ini dikarenakan adanya perbedaan referensi

koordinat (0,0), untuk penggambaran di komputer koordinat (0,0) terletak di pojok kiri atas. Langkah pertama penggambaran sinyal RZ adalah dengan menarik garis setinggi*t* menuju koordinat (0,0) pada penggambaran sinyal RZ pada umumnya. Setelah itu dilihat kondisi dari bit yang akan kita gambar, misalnya, bit ‘1’ maka penggambarannya sebagai berikut :

y 0,0

*a1*

t

*b1*

*b2*

*a2*

**1**

Gambar 9. Penggambaran bit ‘1’ pada komputer

Setelah menuju titik setinggi*t* dari koordinat (0,0) penggambaran sinyal RZ pada umumnya, dilanjutkan dengan membuat garis sepanjang *b1* kearah sumbu (0,0) penggambaran sinyal RZ dengan komputer, setelah itu, dari ujung garis *b1,* buat garis sepanjang garis *a1* ke arah kanan, selanjutnya dari ujung garis *a1* buat garis sepanjang *b2* ke arah bawah (berlawanan dengan arah garis *b1* / menuju koordinat (0,0) penggambaran sinyal RZ pada umumnya), setelah itu tarik garis sepanjang *a2* dengan arah sesuai dengan garis *a1*.

Untuk menggambarkan bit ‘0’ sinyal RZ pada komputer mempunyai cara yang sama dengan penggambaran sinyal RZ dengan bit’1’. Yang membedakan hanyalahbentuk dari sinyal RZ bit ‘0’ yang berlawanan arah dengan bit ‘1’.

# Delphi 7

Delphi merupakan perangkat pengembangan aplikasi yang sangat terkenal dilingkungan Windows.Dengan Delphi, dapat dibangun berbagai aplikasi

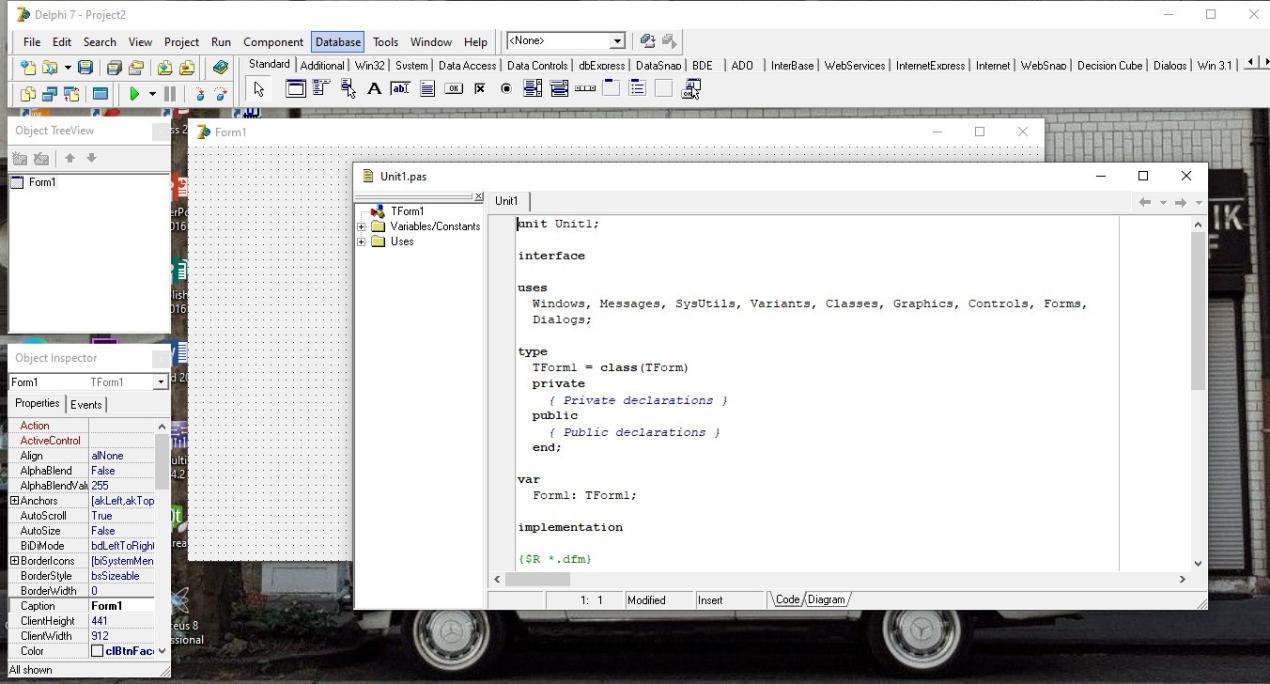
Windows (game, multimedia, database, dll) dengan cepat dan mudah karena menggunakan pendekatan visual yaitu tanpa banyak menuliskan kode. Delphi menggunakan bahasa object Pascal sebagai Bahasa dasar. Jika Bahasa Pascal telah dikuasai maka akan dengan mudah memahami program Delphi.

Langkah-langkah pemrograman visual pada Delphi antara lain:

* Merancang antarmuka (*form* dan komponen pendukungnya) secara visual
* Menuliskan kode untuk melakukan tindakan tertentu
* Mengkompilasikode Pascal dan *form* ke dalam bentuk file yang dapat dieksekus

# IDE Delphi

IDE (*Integrated Development Environment*) adalah lingkungan dimana semua *tool* yang diperlukan untuk merancang, menjalankan, dan menguji sebuah aplikasi disajikan dan terhubung dengan baik sehingga memudahkan dalam pengembangan program. IDE Delphi dibagi menjadi 8 bagian utama yaitu *main menu*, *toolbar/speedbar*,*component palette*, *form designer*, *code editor*, *code explorer*, *object inspector*, dan *object tree view*.



# Main Menu

Gambar Tampilan Jendela Delphi



*Main menu* merupakan menu pilihan utama yang berisi perintah- perintah yang diperlukan selama melakukan pemrograman. *Main menu* terbagi dalam beberapa sub menu sesuai dengan fungsinya.

# Toolbar/Speedbar



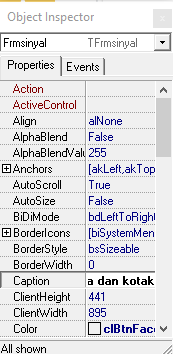
*Toolbar* adalah sekumpulan tombol yang tidak lain adalah penganti beberapa item menu yang sering digunakan. Biasanya yang tersedia pada *toolbar* adalah perintah- perintah (item menu) yang sering digunakan dalam proses pembuatan program aplikasi.

# Component Palette



*Component palette* dalah *tool* yang berupa kumpulan *tab* (*page control*), dimana setiap *tab* memuat berbagai tombol komponen (VCL / *Visual Component Library*) yang dapat diletakkan pada *form* dan sebagai *interface* program aplikasi.*Tab* tersebut diantaranya adalah *Standard*, *Additional*, *Win32*, dan lain-lain.

# Object Inspector



*Object inspector* digunakan untuk mengubah property atau karakteristik dari suatu komponen. Terdiriatas 2 *tab* yaitu :

* + *Properties*

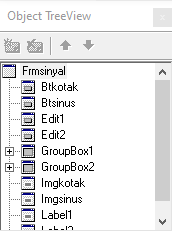
Digunakan untuk menentukan setting suatu objek. Satu objek memiliki beberapa properti yang dapat diatur langsung dari object inspector maupun melalui kode program. Seting ini mempengaruhi cara kerja objek tersebut saat aplikasi dijalankan.

* + *Event*

Merupakan bagian yang dapat diisi dengan kode program tertentu yang berfungsi untuk menangani kejadian-kejadian (berupa sebuah *procedure*) yang dapat direspon oleh sebuah komponen.*Event* adalah peristiwa atau kejadian yang diterima oleh suatuobjek, misal: *click*, *drag*, dan lain-lain. *Event* yang diterima objek akan memicu

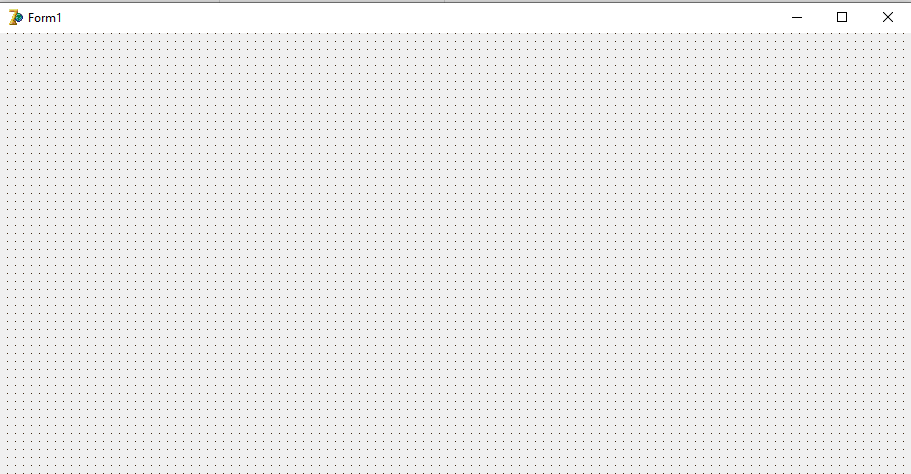
Delphi menjalankan kode program yang ada didalamnya. Misalnya ingin sesuatu dikerjakan pada saat *form* ditutup, makauntukmenyatakantindakantersebut (berupa sebuah *procedure*) menggunakan *OnClose*.

# Object Tree View



*Object tree view* digunakan untuk melihat kontrol-kontrol apa saja yang berada di dalam form tersebut secara hierarki seperti Windows Explorer.

# Form

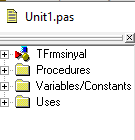


Jendela ini digunakan untuk merancang dan mendesain user interface. Di sini pula kontrol-kontrol komponen dari komponen palet diletakkan/ditempelkan.

# Code Editor

*Code editor* merupakan tempat untuk menuliskan kode program menggunakan Bahasa *object Pascal*. Disini tidak perludituliskan seluruh kode sumber karena Delphi telah menyediakan kerangka penulisan sebuah program.

# Code Explorer

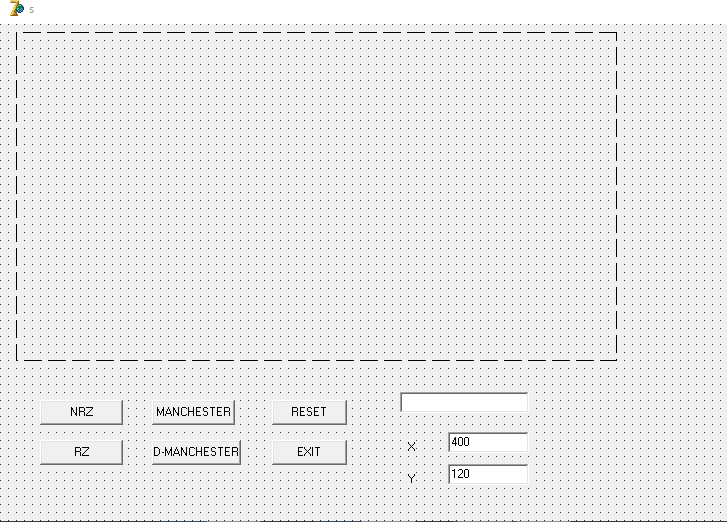


Digunakan untuk memudahkan berpindah antar file unit di dalam jendela *code editor*. *Code explorer* berisi daftar yang menampilkan semua *tipe, class, properti, method*, variabel global, rutin global yang telah didefinisikan di dalam unit. Saat memilih sebuah item dalam *code*

*explorer*, kursor akan berpindah menuju implementasi dari item yang dipilih di dalam code editor.

# Prosedur

1. Buka program Delphi 7
2. Tambahkan pada form satu buah image, enam buah button, dua buah label,dan tiga buah edit. Selanjutnya atur tata letak komponen tersebut sesuai dengan gambar dibawah ini.



Gambar Tata letak komponen

1. Ubah property komponen – komponen tersebut menjadi seperti pada tabel 1

|  |  |  |
| --- | --- | --- |
| Komponen | Property | Setting |
| Form | Caption | Pengkodean RZ |
| Name | frmsinyal |
| Button1 | Caption | NRZ-L |
| Name | Btrzl |
| Button2 | Caption | RZ |
| Name | btrz |

|  |  |  |
| --- | --- | --- |
| Button3 | Caption | Manchester |
|  | Name | btmanch |
| Button4 | Caption | D-Manchester |
|  | Name | btdmanch |
| Button5 | Caption | Reset |
|  | Name | bteset |
| Button6 | Caption | Exit |
|  | Name | btexit |
| Label 1 | Caption | X |
| Label 2 | Caption | Y |
| Edit 1 | Caption | Kosongkan |
| Name | Edbiner |
| Edit 2 | Text | 400 |
| Name | Edsumbux |
| Edit 3 | Text | 120 |
| Name | Edsumbuy |

# Flowchart

* 1. **Listing program**

unit Unit14;

interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls;

type

Tfrmrz = class(TForm) Imgtampil: TImage; btnrz: TButton; btnreset: TButton; btnz: TButton; btmanc: TButton; btdmanc: TButton; btexit: TButton; Edit1: TEdit; edsumbux: TEdit; edsumbuy: TEdit; Label1: TLabel; Label2: TLabel;

procedure btexitClick(Sender: TObject); procedure btnrzClick(Sender: TObject); procedure btmancClick(Sender: TObject); procedure btdmancClick(Sender: TObject); procedure btnresetClick(Sender: TObject);

private

{ Private declarations } public

{ Public declarations } end;

var

frmrz: Tfrmrz; implementation

{$R \*.dfm}

procedure sumbu (ax,ay,b,c:integer); Begin frmrz.imgtampil.Canvas.MoveTo(ax,ay);

frmrz.imgtampil.Canvas.LineTo(ax+b,ay); frmrz.imgtampil.Canvas.MoveTo(ax,ay); frmrz.imgtampil.Canvas.LineTo(ax,ay-c); frmrz.imgtampil.Canvas.MoveTo(ax,ay); frmrz.imgtampil.Canvas.LineTo(ax,ay+c); end;

//RZ

procedure rz\_1(x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clRed; frmrz.imgtampil.Canvas.MoveTo(40+x,150); frmrz.imgtampil.Canvas.LineTo(40+x,100); frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(70+y,100); frmrz.imgtampil.Canvas.MoveTo(70+y,100); frmrz.imgtampil.Canvas.lineto(70+y,150); frmrz.imgtampil.Canvas.MoveTo(70+y,150); frmrz.imgtampil.Canvas.LineTo(100+y,150); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure rz\_0(x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clRed; frmrz.imgtampil.Canvas.MoveTo(40+x,150); frmrz.imgtampil.Canvas.LineTo(40+x,200);

frmrz.imgtampil.Canvas.MoveTo(40+x,200); frmrz.imgtampil.Canvas.LineTo(70+y,200); frmrz.imgtampil.Canvas.MoveTo(70+y,200); frmrz.imgtampil.Canvas.lineto(70+y,150); frmrz.imgtampil.Canvas.MoveTo(70+y,150); frmrz.imgtampil.Canvas.LineTo(100+y,150); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure Tfrmrz.BtrzClick(Sender: TObject); var i:integer;

data:string; sx,sy,x,y:integer;

begin

data:='10101010'; x:=0;

y:=0;

for i:=1 to 8 do begin

if data[i]='1' then begin; rz\_1(x,y);

end else if data[i]='0' then begin

rz\_0 (x,y); end; y:=y+60; x:=x+60;

end; x:=0;

y:=0;

x:= strtoint(frmrz.edsumbux.Text); y:= strtoint(frmrz.edsumbuy.Text); sumbu(40,150,500,y);

for sx:=1 to 384 do begin sy:=y;

end;

end;

//NRZ-L

procedure garis0(x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(100+y,100); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure garis1(x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+y,200); frmrz.imgtampil.Canvas.LineTo(100+y,200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure garistegak (x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(40+x,200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure Tfrmrz.btnrzClick(Sender: TObject); var i:integer;

data:string; sx,sy,x,y:integer;

begin data:=frmrz.Edit1.Text; x:=0;

y:=0;

for i:=1 to length(frmrz.Edit1.Text) do begin

if data[i]='1' then begin

garis1(x,y); end;

if data[i]='0' then begin

garis0(x,y); end;

if data[i]<>data[i-1] then begin

garistegak(x,y); end;

y:=y+60; x:=x+60;

end; x:=0;

y:=0;

x:= strtoint(frmrz.edsumbux.Text); y:= strtoint(frmrz.edsumbuy.Text); sumbu(40,150,500,y);

for sx:=1 to 500 do

begin

end; end;

sy:=y;

//Manchester

procedure sumbumanc (ax,ay,b,c:integer); Begin frmrz.imgtampil.Canvas.MoveTo(ax,ay); frmrz.imgtampil.Canvas.LineTo(ax+b,ay); frmrz.imgtampil.Canvas.MoveTo(ax,ay); frmrz.imgtampil.Canvas.LineTo(ax,ay-c);

frmrz.imgtampil.Canvas.MoveTo(ax,ay); frmrz.imgtampil.Canvas.LineTo(ax,ay+c); end;

procedure garismanc0(x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(70+y,100); frmrz.imgtampil.Canvas.LineTo(70+y,200); frmrz.imgtampil.Canvas.MoveTo(70+y,200); frmrz.imgtampil.Canvas.LineTo(100+x,200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure garismanc1 (x,y:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+x,200); frmrz.imgtampil.Canvas.LineTo(70+y,200); frmrz.imgtampil.Canvas.MoveTo(70+y,200); frmrz.imgtampil.Canvas.LineTo(70+y,100); frmrz.imgtampil.Canvas.MoveTo(70+y,100); frmrz.imgtampil.Canvas.LineTo(100+y,100); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure garistegakmanc (x,y:integer); begin frmrz.imgtampil.Canvas.Pen.Color:=clblue;

frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(40+x,200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure garistegakmanc1 (x,y:integer);

begin frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo(40+x,100); frmrz.imgtampil.Canvas.LineTo(40+x,150); frmrz.imgtampil.Canvas.Pen.Color:=clBlack; end;

procedure Tfrmrz.btmancClick(Sender: TObject); var i:integer;

data:string; sx,sy,x,y:integer;

begin data:=frmrz.Edit1.Text; x:=0;

y:=0;

for i:=1 to length(frmrz.Edit1.Text) do begin

if data[i]='1' then begin

garismanc1(x,y); end;

if data[i]='0' then begin

garismanc0(x,y); end;

if data[i]=data[i-1] then begin

garistegakmanc(x,y); end;

y:=y+60; x:=x+60;

end; x:=0;

y:=0;

x:= strtoint(frmrz.edsumbux.Text); y:= strtoint(frmrz.edsumbuy.Text); sumbu(40,150,500,y);

for sx:=1 to 500 do

begin

end; end;

sy:=y;

//D-Manchester

procedure dmncstr\_1(ax,ay,a,b,i:integer); begin frmrz.imgtampil.Canvas.Pen.Color:=clblue;

frmrz.imgtampil.Canvas.MoveTo(ax+(i-1)\*b+((i-1)\*b),200); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+b,200); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+b,100); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+2\*b,100); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure dmncstr\_0(ax,ay,a,b,i:integer); begin frmrz.imgtampil.Canvas.Pen.Color:=clblue;

frmrz.imgtampil.Canvas.MoveTo(ax+(i-1)\*b+((i-1)\*b),100); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+b,100); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+b,200); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b))+2\*b,200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure dtegak(ax,ay,a,b,i:integer); begin

frmrz.imgtampil.Canvas.Pen.Color:=clblue; frmrz.imgtampil.Canvas.MoveTo((ax+(i-1)\*b+((i-1)\*b)),100); frmrz.imgtampil.Canvas.LineTo((ax+(i-1)\*b+((i-1)\*b)),200); frmrz.imgtampil.Canvas.Pen.Color:=clBlack;

end;

procedure Tfrmrz.btdmancClick(Sender: TObject); var

i:integer; sx,sy,x,y:integer; tnd,ax,ay,a,b,n:integer; edbiner:string;

begin edbiner:=frmrz.Edit1.Text; ax:=40;

ay:=150; a:=30;

b:=30;

tnd:=1; x:=0;

y:=0;

x:= strtoint(frmrz.edsumbux.Text); y:= strtoint(frmrz.edsumbuy.Text); sumbu(40,150,600,y);

for sx:=1 to 500 do begin

sy:=y; end;

for n:=1 to length(edbiner) do begin

if edbiner[n]='1' then if tnd = 0 then

begin dmncstr\_1(ax,ay,b,a,n); tnd:= 1;

end

else

begin dmncstr\_0(ax,ay,b,a,n); tnd:=0;

end else

if tnd = 0 then

begin dtegak(ax,ay,b,a,n);

dmncstr\_0(ax,ay,b,a,n); end

else

begin dtegak(ax,ay,b,a,n); dmncstr\_1(ax,ay,b,a,n);

end end;

end;

procedure Tfrmrz.btexitClick(Sender: TObject); begin

close end;

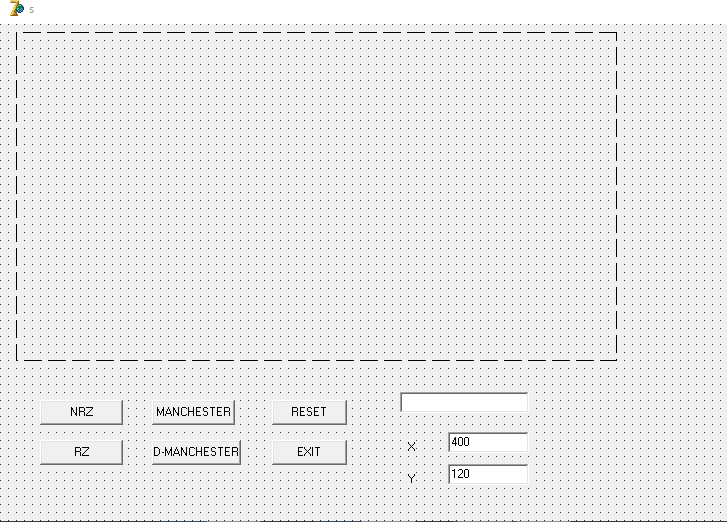
procedure Tfrmrz.btnresetClick(Sender: TObject); begin

imgtampil.Picture:=nil; imgtampil.Refresh;

end; end.

# Tampilan Hasil Program

1. **Tampilan Form NRZ-L, NRZ-I, RZ, MANCHESTER, D-MANCHESTER,AMI**



# Tampilan data hasil pratikum

|  |  |
| --- | --- |
| **Kode Biner** | **Hasil simulasi** |
| **10101010** | **1. RZ** |

|  |  |
| --- | --- |
|  | 1. **NRZ-L**      1. **MANCHESTER** |

|  |  |
| --- | --- |
|  | **4. D- MANCHESTER** |

|  |  |
| --- | --- |
|  |  |
| **01011001** | 1. **RZ**      1. **NRZ-L** |

|  |  |
| --- | --- |
|  | 1. **MANCHESTER**      1. **D- MANCHESTER** |

|  |  |
| --- | --- |
|  |  |
| **10010101** | 1. **RZ**      1. **NRZ-L** |

|  |  |
| --- | --- |
|  | **3. MANCHESTER** |

|  |  |
| --- | --- |
|  | **4. D- MANCHESTER** |
| **11101101** | **1. RZ** |

|  |  |
| --- | --- |
|  | 1. **NRZ-L**      1. **MANCHESTER** |

|  |  |
| --- | --- |
|  | **4. D- MANCHESTER** |
| **00010100** | **1. RZ** |

|  |  |
| --- | --- |
|  | 1. **NRZ-L**      1. **MANCHESTER** |

|  |  |
| --- | --- |
|  | **4. D- MANCHESTER** |

* 1. **Analisa**

# RZ

Sinyal RZ memiliki format pengkodean yang selalu menuju ke level 0 pada setengah periodenya. Dapat dilihat dari hasil pratikum yang menunjukkan bahwa biner “0’ diwakili oleh perubahan level dari negative ke 0 dan untuk biner “1” diwakili oleh perubahan level dari positive ke 0.

# NRZ-L

Sinyal NRZ-L memiliki format pengkodean yang tegangan positif diwakili dengan biner “0 ” dan tegangan negatif diwakili dengan biner “1” dapat dilihat dari hasil pratikum.

# Manchester

Berdasarkan gambar hasil pratikum untuk sinyal Manchester akan terjadi suatu transisi pada setengah periode, dimana biner “1” akan diwakili dengan transisi rendah ke tinggi dan biner “0” akan diwakili dengan transisi tinggi ke rendah.

# D-Manchester

Pada sinyal D-Manchester, biner “1 ” akan diwakili dengan transisi diawal periode dan untuk biner “0 ” akan diwakili dengan tidak adanya transisi diawal periode suatu bit.

# Kesimpulan

Berdasarkan hasil pratikum yang telah dilakukan dapat disimpulkan :

1. Data digital adalah data yang memiliki rangkaian nilai yang berbeda dan memiliki karakteristik tersendiri, dan sinyal digital adalah hasil teknologi yang dapat mengubah sinyal menjadi kombinasi 0 dan 1.
2. Pengkodean saluran kembali ke nol (RZ) menggunakan level –V dan +V yang berpusat pada biner data dari bit bergerak. Biner 0 diwakili oleh transisi dari –V ke 0 V, sedangkan biner 1 diwakili oleh transisi dari 0 V ke V.
3. Pengkodean Manchaster akan terjadi suatu transisi pada setengah periode, dimana biner “1” akan diwakili dengan transisi rendah ke tinggi dan biner “0” akan diwakili dengan transisi tinggi ke rendah.
4. Pengkodean D-Manchaster adalah kode di mana biner 0 diwakili oleh transisi di awal. periode bit dan biner 1 menunjukkan tidak ada transisi pada awal periode bit.