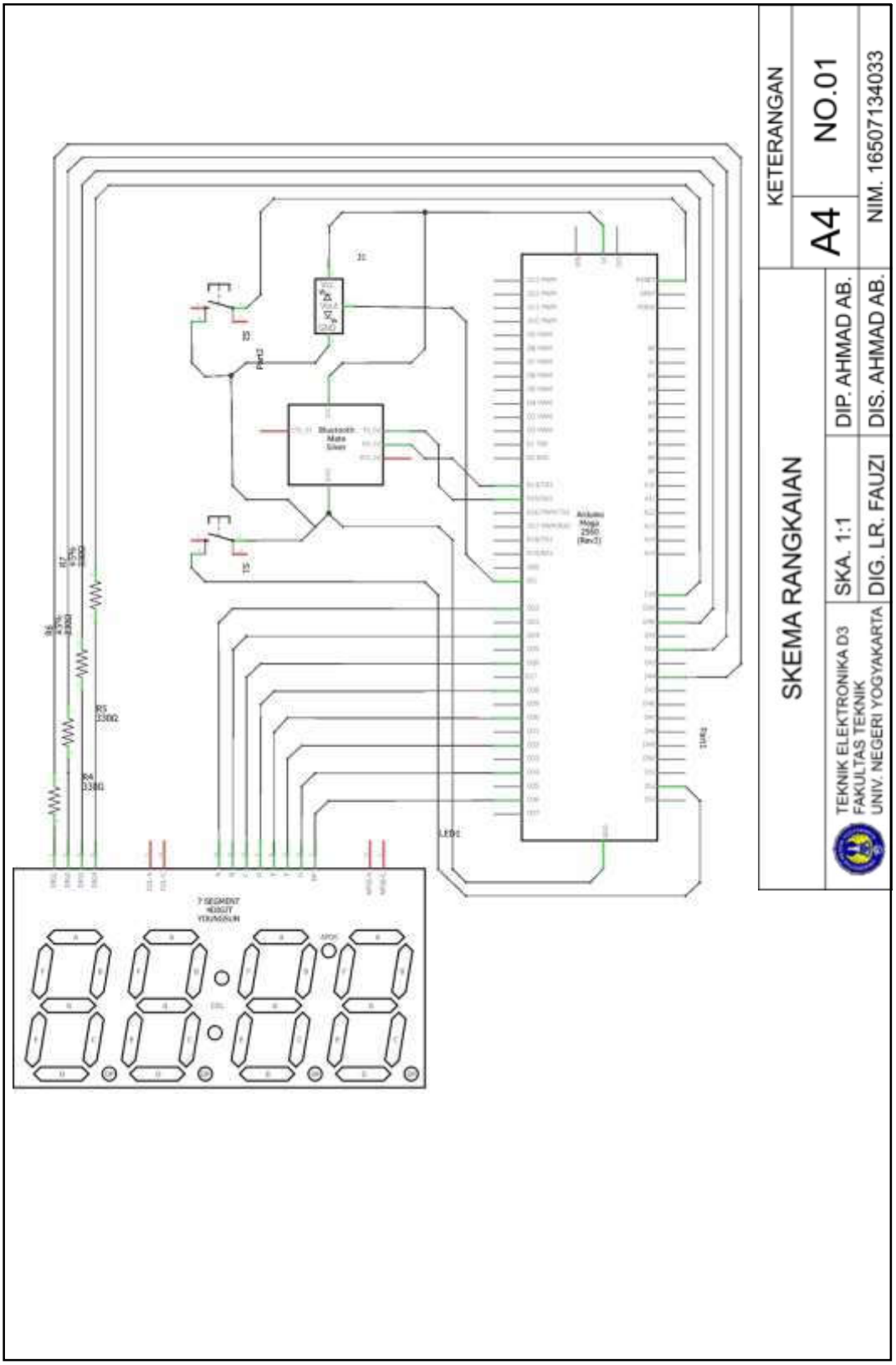
LAMPIRAN

Lampiran 1. Skema Rangkaian



Lampiran 2. *Coding* Arduino

```
#define segment1 38 // pin common untuk seven segment 1 pada Arduino
#define segment2 40 //pin common untuk seven segment 2 pada Arduino
#define segment3 42 //pin common untuk seven segment 2 pada Arduino
#define segment4 44 //pin common untuk seven segment 2 pada Arduino

unsigned int digit[10] = {

  B00000011, // 0
  B10011111, // 1
  B00100101, // 2
  B00001101, // 3
  B10011001, // 4
  B01001001, // 5
  B01000001, // 6
  B00011111, // 7
  B00000001, // 8
  B00001001, // 9
};
byte pinSegment[] = {36,34,32,30,28,26,24,22


const int buttonPin = 52;
int buttonState = 0;
unsigned int persen;
int sensor1=21;
unsigned long proxy_kanan;
long t;
long h;
long i;
unsigned long timeold_ka;
unsigned long timeandro;
unsigned long timeandro1;
long double rpm_ka;
unsigned int ardu;
char data =0 ;

void speed_kanan()
{
```

```
  proxy_kanan++;

}
void rpm_kanan()
{
 if (millis() > t + 300)
  {
  detachInterrupt(2);
  timeold_ka = millis();
  rpm_ka = proxy_kanan*0.0146;
  proxy_kanan = 0;

  attachInterrupt(2, speed_kanan, FALLING);

  t=millis();
  }
}

void setup()
  {
  Serial.begin(9600);
  for(int i=0; i < 8; i++)

   {
      pinMode(pinSegment[i], OUTPUT);
   }

   pinMode(segment1,OUTPUT);
   pinMode(segment2,OUTPUT
   pinMode(segment3,OUTPUT);
   pinMode(segment4,OUTPUT);

   pinMode(sensor1,INPUT_PULLUP);
   attachInterrupt(2, speed_kanan, FALLING);

   proxy_kanan = 0;
   rpm_ka = 0;
   timeold_ka = 0;
   timeandro = 0;
```

```
  for ( byte segment = 0; segment < 8; segment++)
     {
      boolean Status = bitRead(digit[0],segment);
      digitalWrite(pinSegment[segment],Status);
     }

  digitalWrite(segment1, HIGH);
  digitalWrite(segment2, HIGH);
  digitalWrite(segment3, HIGH);
  digitalWrite(segment4, HIGH);
  delay(1000);
  digitalWrite(segment1, LOW);
  digitalWrite(segment2, LOW);
  digitalWrite(segment3, LOW);
  digitalWrite(segment4, LOW);
  delay(500);
  digitalWrite(segment1, HIGH);
  digitalWrite(segment2, HIGH);
  digitalWrite(segment3, HIGH);
  digitalWrite(segment4, HIGH);
  delay(500);
  digitalWrite(segment1, LOW);
  digitalWrite(segment2, LOW);
  digitalWrite(segment3, LOW);
  digitalWrite(segment4, LOW);
  delay(200);
  digitalWrite(segment1, HIGH);
  digitalWrite(segment2, HIGH);
  digitalWrite(segment3, HIGH);
  digitalWrite(segment4, HIGH);
  delay(200);
  digitalWrite(segment1, LOW);
  digitalWrite(segment2, LOW);
  digitalWrite(segment3, LOW);
  digitalWrite(segment4, LOW);
  delay(100);

  pinMode(buttonPin, INPUT);
}
```

```
void loop()
  {
    utama();
    hold();
  }

void utama()
{
    rpm_kanan();
    persen = rpm_ka*10;
    tampilkan(persen);
    ardu = rpm_ka*10;
    blue();
}

void hold()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
    unsigned int ambil = rpm_ka*10;
    noInterrupts();
    tampilkan(ambil);

    if (millis() > i + 300)

    {
     timeandro1 = millis();
       if (ambil > 0){
         Serial.print(ambil);
       }
       if (ambil == 0){
         Serial.print("00");
       }
      i=millis();
    }
  }
}
```

```
void blue()
{
 if (millis() > h + 1000)
  {
  timeandro = millis();
  if (ardu > 0){
  Serial.print(ardu);

  }
  if (ardu == 0){
  Serial.print("00");
  }
  h=millis();
  }
}

void tampilkan(unsigned int nilai)
{
  if (nilai < 10 && nilai < 100)
  {

      digitalWrite(segment1,LOW);

      for ( byte segment = 0; segment < 8; segment++)
       {
        boolean Status = bitRead(digit[0],segment);
        digitalWrite(pinSegment[segment],Status);
       }

    digitalWrite(segment2, HIGH);
    delay(5);

    digitalWrite(segment2, LOW);

     for ( byte segment = 0; segment < 8; segment++)
     {
     boolean Status = bitRead(digit[nilai],segment);
     digitalWrite(pinSegment[segment],Status);
     }
     digitalWrite(segment1,HIGH);
```

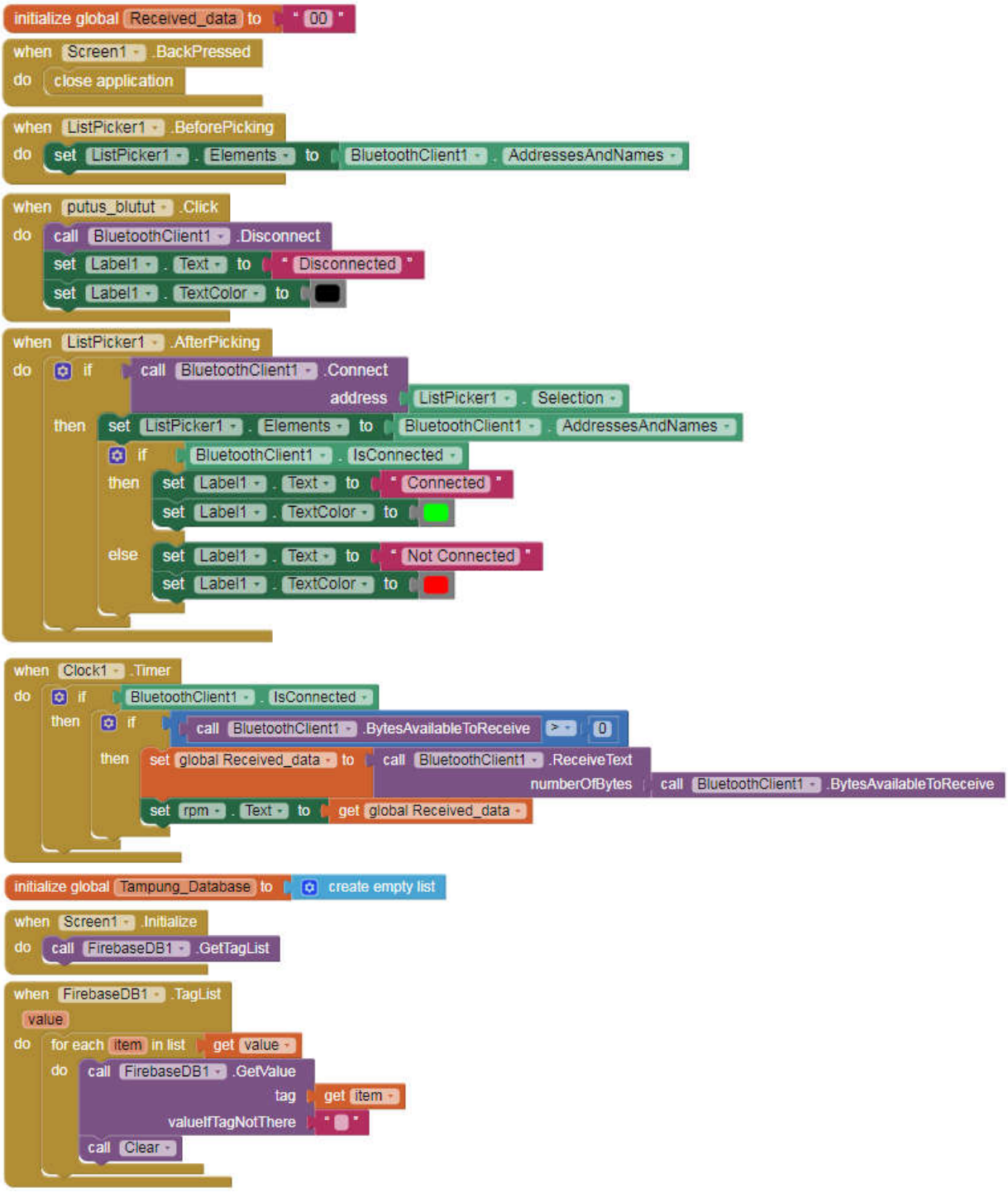
```
    delay(5);
  }
  else if ( nilai > 9 && nilai < 100)
 {
  byte digit1 = nilai/10;
  byte digit2 = nilai-(digit1*10);

     digitalWrite(segment1,LOW);

     for ( byte segment = 0; segment < 8; segment++)
      {
        boolean Status = bitRead(digit[digit1],segment);
        digitalWrite(pinSegment[segment],Status);
      }

   digitalWrite(segment2, HIGH);
   delay(5);

   digitalWrite(segment2, LOW);

    for ( byte segment = 0; segment < 8; segment++)
      {
        boolean Status = bitRead(digit[digit2],segment);
        digitalWrite(pinSegment[segment],Status);
      }

    digitalWrite(segment1,HIGH);
    delay(5);
 }

 else if ( nilai > 99 && nilai < 1000)
 {
  unsigned int bilangan1 = nilai/100;
  unsigned int bilangan22 = nilai-(bilangan1*100);
  unsigned int bilangan2 = bilangan22/10;
  unsigned int bilangan3 = bilangan22-(bilangan2*10);

  digitalWrite(segment1,LOW);
  digitalWrite(segment2,LOW);
```

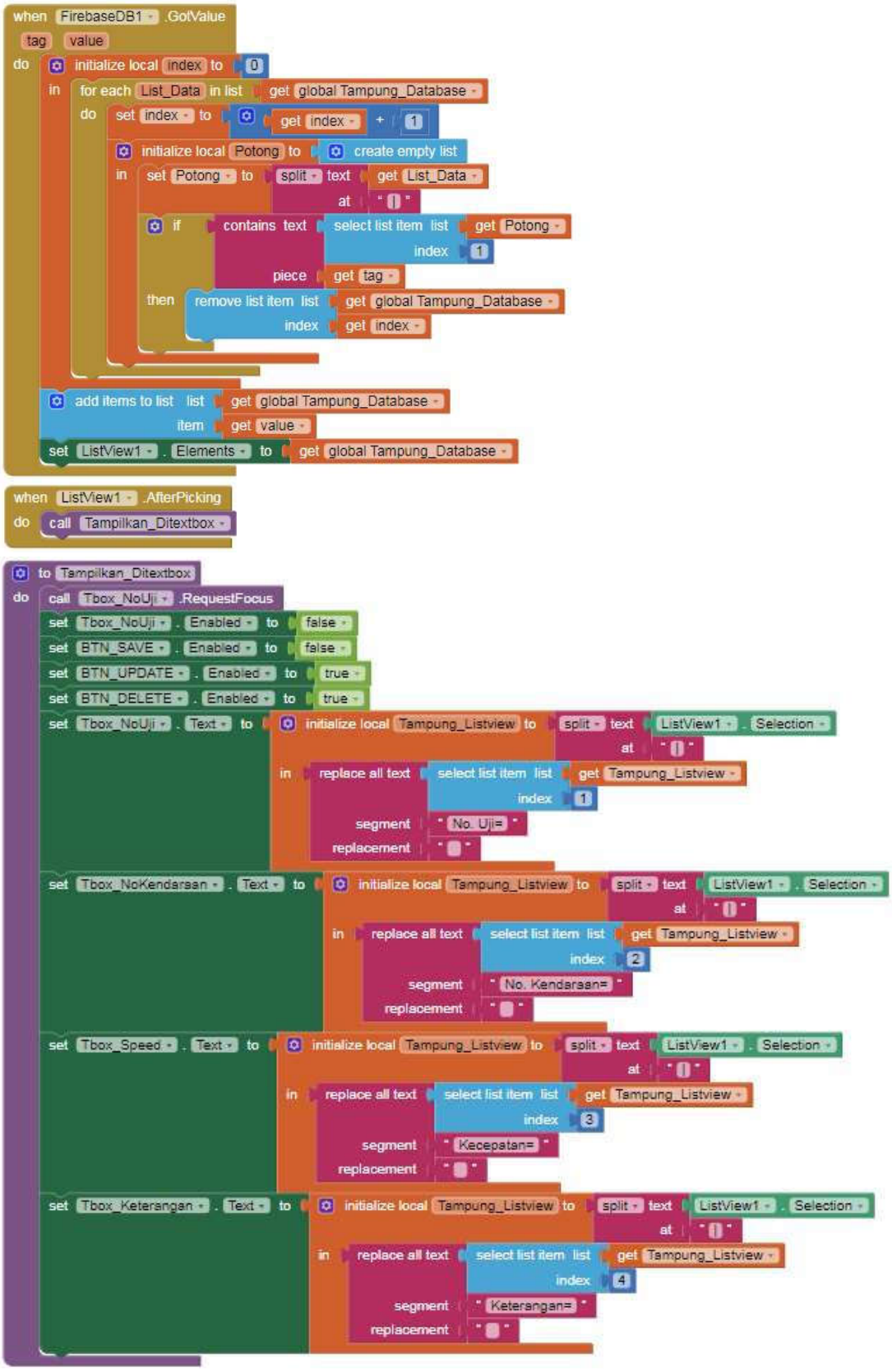
```
      for ( byte segment = 0; segment < 8; segment++)
       {
        boolean Status = bitRead(digit[bilangan1],segment);

        digitalWrite(pinSegment[segment],Status);
       }

    digitalWrite(segment3, HIGH);
    delay(5);

    digitalWrite(segment3, LOW);

     for ( byte segment = 0; segment < 8; segment++)
      {
        boolean Status = bitRead(digit[bilangan2],segment);

        digitalWrite(pinSegment[segment],Status);
      }

     digitalWrite(segment2,HIGH);
     delay(5);

     digitalWrite(segment2, LOW);

     for ( byte segment = 0; segment < 8; segment++)
      {
        boolean Status = bitRead(digit[bilangan3],segment);
        digitalWrite(pinSegment[segment],Status);
      }

     digitalWrite(segment1,HIGH);
     delay(5);
  }

  else if ( nilai > 999 && nilai < 10000)
  {
   unsigned int angka1= nilai/1000;
   unsigned int angka11 = nilai-(angka1*1000);
   unsigned int angka2 = angka11/100;
   unsigned int angka22 = angka11-(angka2*100);
```

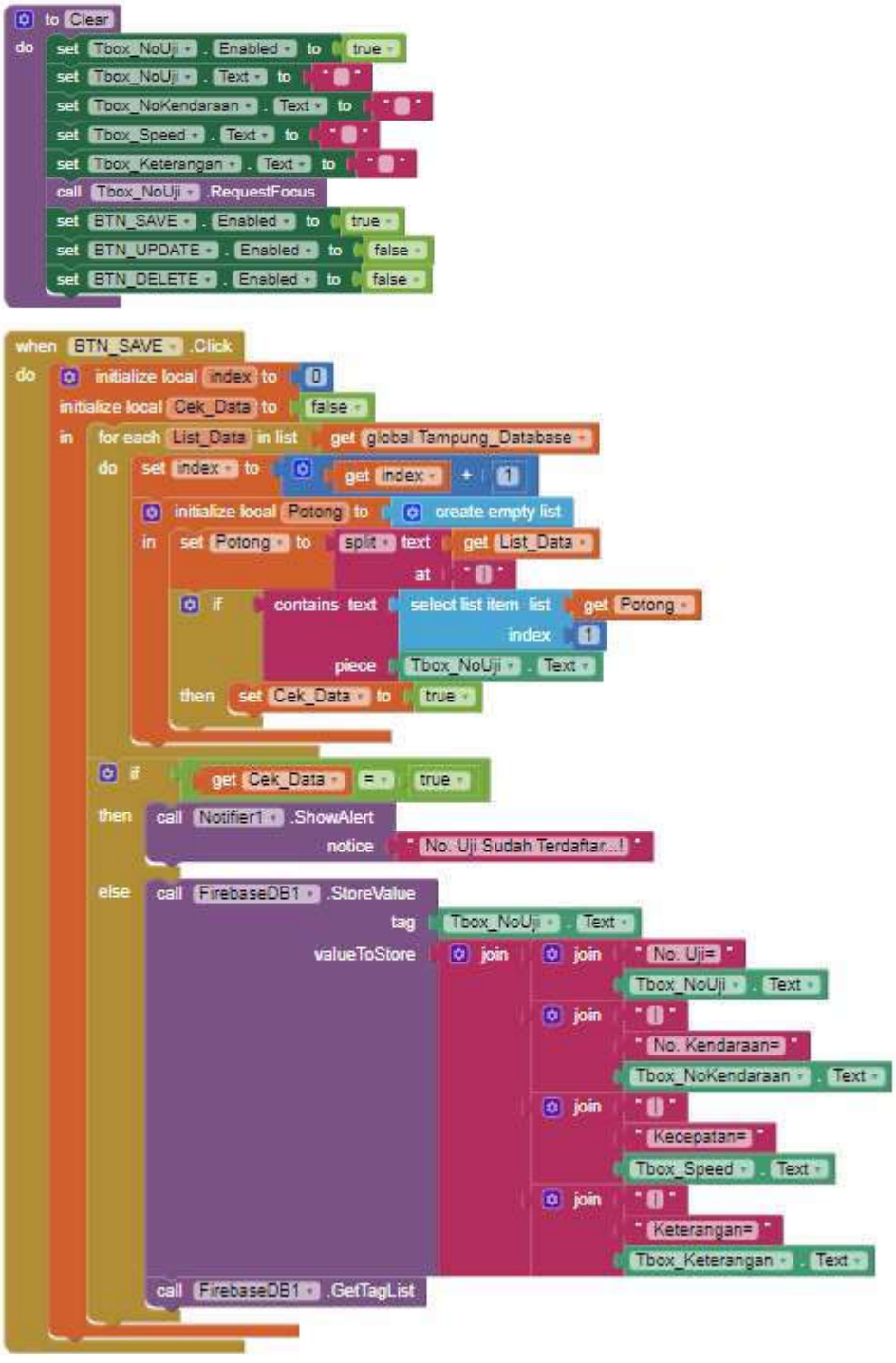
```
unsigned int angka3 = angka22/10;
unsigned int angka4 = angka22-(angka3*10);

   digitalWrite(segment1,LOW);
   digitalWrite(segment2,LOW);
   digitalWrite(segment3,LOW);

   for ( byte segment = 0; segment < 8; segment++)
    {
     boolean Status = bitRead(digit[angka1],segment);
     digitalWrite(pinSegment[segment],Status);
    }

 digitalWrite(segment4, HIGH);
 delay(5);

  digitalWrite(segment4, LOW);

  for ( byte segment = 0; segment < 8; segment++)
    {
     boolean Status = bitRead(digit[angka2],segment);
     digitalWrite(pinSegment[segment],Status);
    }

  digitalWrite(segment3,HIGH);
  delay(5);

  digitalWrite(segment3, LOW);

  for ( byte segment = 0; segment < 8; segment++)
    {
     boolean Status = bitRead(digit[angka3],segment);
     digitalWrite(pinSegment[segment],Status);
    }

  digitalWrite(segment2,HIGH);
  delay(5);

  digitalWrite(segment2, LOW);
```

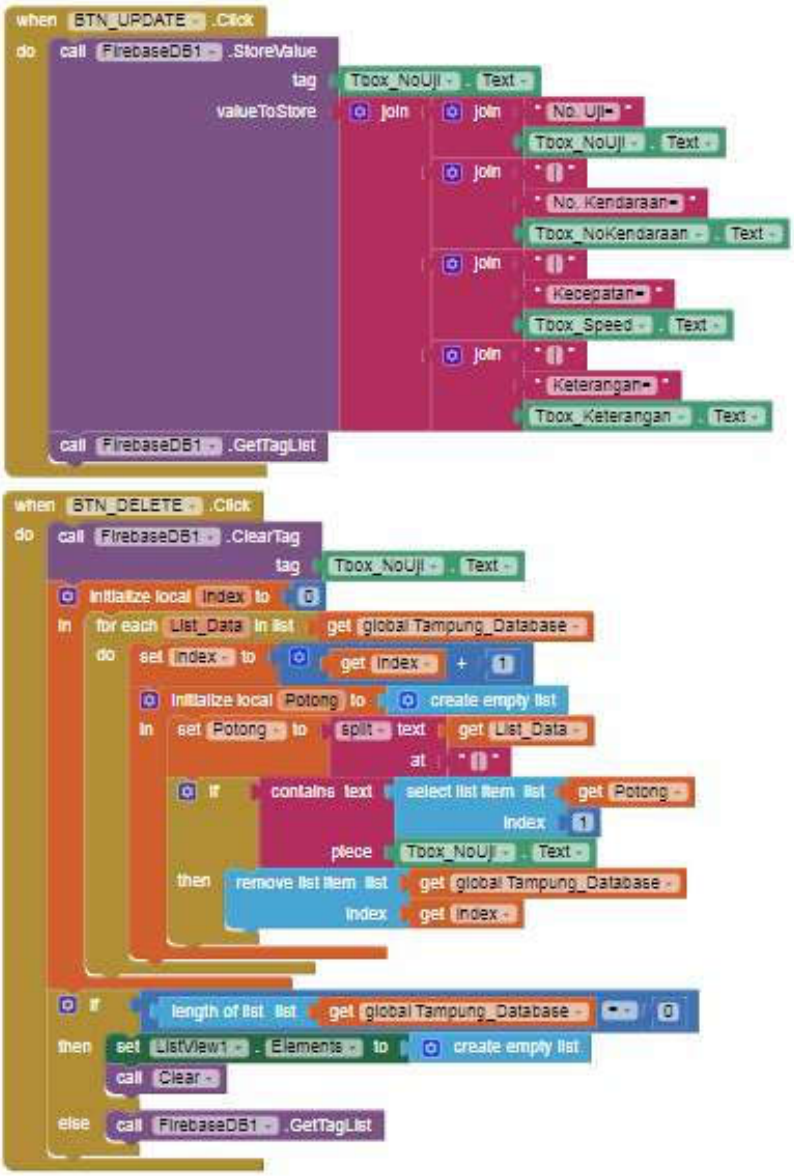
```
    for ( byte segment = 0; segment < 8; segment++)
      {
       boolean Status = bitRead(digit[angka4],segment);
       digitalWrite(pinSegment[segment],Status);
      }

    digitalWrite(segment1,HIGH);
    delay(5);
  }
}
```

Lampiran 3. *Coding* Aplikasi




when FirebaseDB1 .GotValue
tag value
do initialize local index to 0
in for each List_Data in list get global Tampung_Database
do set index to get index + 1
initialize local Potong to create empty list
in set Potong to split text get List_Data
at " [] "
if contains text select list item list get Potong
index 1
piece get tag
then remove list item list get global Tampung_Database
index get index
add items to list list get global Tampung_Database
item get value
set ListView1 . Elements to get global Tampung_Database
when ListView1 .AfterPicking
do call Tampilkan_Ditextbox
to Tampilkan_Ditextbox
do call Tbox_NoUji .RequestFocus
set Tbox_NoUji . Enabled to false
set BTN_SAVE . Enabled to false
set BTN_UPDATE . Enabled to true
set BTN_DELETE . Enabled to true
set Tbox_NoUji . Text to initialize local Tampung_Listview to split text ListView1 . Selection
at " [] "
in replace all text select list item list get Tampung_Listview
index 1
segment " No. Uji= "
replacement " "
set Tbox_NoKendaraan . Text to initialize local Tampung_Listview to split text ListView1 . Selection
at " [] "
in replace all text select list item list get Tampung_Listview
index 2
segment " No. Kendaraan= "
replacement " "
set Tbox_Speed . Text to initialize local Tampung_Listview to split text ListView1 . Selection
at " [] "
in replace all text select list item list get Tampung_Listview
index 3
segment " Kecepatan= "
replacement " "
set Tbox_Keterangan . Text to initialize local Tampung_Listview to split text ListView1 . Selection
at " [] "
in replace all text select list item list get Tampung_Listview
index 4
segment " Keterangan= "
replacement " "

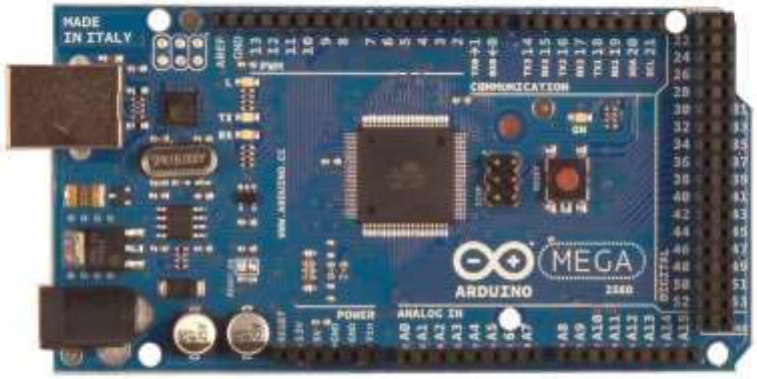

to Clear
do
set Tbox_NoUji . Enabled to true
set Tbox_NoUji . Text to ""
set Tbox_NoKendaraan . Text to ""
set Tbox_Speed . Text to ""
set Tbox_Keterangan . Text to ""
call Tbox_NoUji .RequestFocus
set BTN_SAVE . Enabled to true
set BTN_UPDATE . Enabled to false
set BTN_DELETE . Enabled to false
when BTN_SAVE .Click
do
initialize local index to 0
initialize local Cek_Data to false
in
for each List_Data in list get global Tampung_Database
do
set index to get index + 1
initialize local Potong to create empty list
in
set Potong to split text get List_Data
at "|"
if contains text select list item list get Potong
index 1
piece Tbox_NoUji . Text
then set Cek_Data to true
if get Cek_Data = true
then call Notifier1 .ShowAlert
notice "No. Uji Sudah Terdaftar...!"
else call FirebaseDB1 .StoreValue
tag Tbox_NoUji . Text
valueToStore join join "No. Uji=" Tbox_NoUji . Text
join "|" "No. Kendaraan=" Tbox_NoKendaraan . Text
join "|" "Kecepatan=" Tbox_Speed . Text
join "|" "Keterangan=" Tbox_Keterangan . Text
call FirebaseDB1 .GetTagList

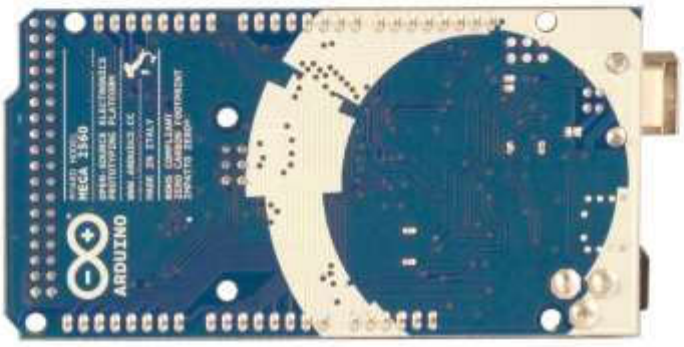

when BTN_UPDATE .Click
do call FirebaseDB1 .StoreValue
tag Tbox_NoUji . Text
valueToStore join join "No. Uji-" Tbox_NoUji . Text
join "|" "No. Kendaraan-" Tbox_NoKendaraan . Text
join "|" "Kecepatan-" Tbox_Speed . Text
join "|" "Keterangan-" Tbox_Keterangan . Text
call FirebaseDB1 .GetTagList
when BTN_DELETE .Click
do call FirebaseDB1 .ClearTag
tag Tbox_NoUji . Text
initialize local index to 0
in for each List_Data in list get global Tampung_Database
do set index to get index + 1
initialize local Potong to create empty list
in set Potong to split text get List_Data
at "|"
if contains text select list item list get Potong
index 1
piece Tbox_NoUji . Text
then remove list item list get global Tampung_Database
index get index
if length of list list get global Tampung_Database = 0
then set ListView1 . Elements to create empty list
call Clear
else call FirebaseDB1 .GetTagList

Lampiran 4. *Datasheet* Arduino Mega 2560

ARDUINO

Arduino Mega 2560 Datasheet





Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

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

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

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

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

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

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH

value, the LED is on, when the pin is LOW, it's off.

- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

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

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It

communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

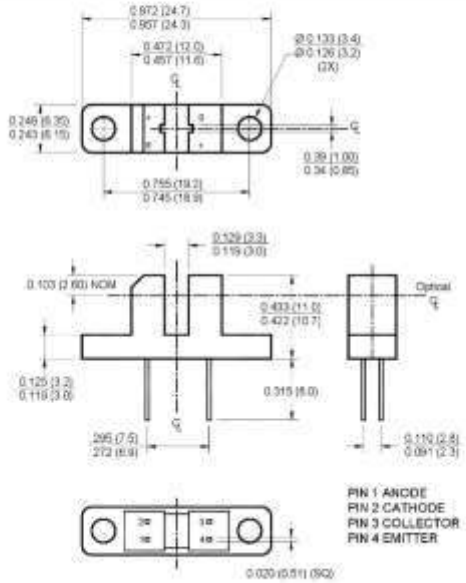
Lampiran 5. *Datasheet* Sensor Optocoupler

FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

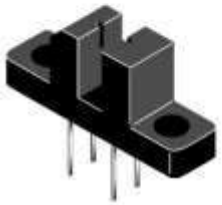
H21A4 H21A5 H21A6

PACKAGE DIMENSIONS

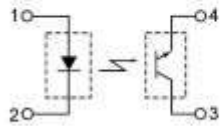


NOTES:

1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.



SCHEMATIC



DESCRIPTION

The H21A series are gallium arsenide infrared emitting diode coupled with a silicon photodarlington in a plastic housing. The packaging system is designed to optimize the mechanical resolution, coupling efficiency, ambient light rejection, cost and reliability. The gap in the housing provides a means of interrupting the signal with an opaque material, switching the output from an "ON" to an "OFF" state.

FEATURES

- Opaque housing
- Low cost
- .035" apertures
- High $I_{C(ON)}$

FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

H21A4 H21A5 H21A6

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

Parameter	Symbol	Rating	Unit
Operating Temperature	T_{OPR}	-55 to +100	°C
Storage Temperature	T_{STG}	-55 to +100	°C
Soldering Temperature (Iron)[2,3 and 4]	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)[2 and 3]	T_{SOL-F}	260 for 10 sec	°C
INPUT (EMITTER)			
Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	6	V
Power Dissipation [1]	P_D	100	mW
OUTPUT (SENSOR)			
Collector to Emitter Voltage	V_{CEO}	55	V
Emitter to Collector Voltage	V_{ECO}	4.5	V
Collector Current	I_C	20	mA
Power Dissipation (T_C = 25°C)[1]	P_D	150	mW

NOTE:
1. Derate power dissipation linearly 1.67 mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron tip 1/16" (1.6mm) minimum from housing.

ELECTRICAL / OPTICAL CHARACTERISTICS (T_A =25°C) (All measurements made under pulse conditions)

PARAMETER	TEST CONDITIONS	SYMBOL	DEVICES	MIN	TYP	MAX	UNITS
INPUT (EMITTER)							
Forward Voltage	I_F = 60 mA	V_F	All	—	—	1.7	V
Reverse Breakdown Voltage	I_R = 10 μA	V_R	All	6.0	—	—	μA
Reverse Leakage Current	V_R = 3 V	I_R	All	—	—	1.0	μA
OUTPUT (SENSOR)							
Emitter to Collector Breakdown	I_F = 100 μA, Ee = 0	BV_{ECO}	All	6.0	—	—	V
Collector to Emitter Breakdown	I_C = 1 mA, Ee = 0	BV_{CEO}	All	55	—	—	V
Collector to Emitter Leakage	V_{CE} = 45 V, Ee = 0	I_{CEO}	All	—	—	100	nA
COUPLED							
On-State Collector Current	I_F = 5 mA, V_{CE} = 5 V	$I_{C(ON)}$	H21A4	0.15	—	—	mA
			H21A5	0.30	—	—	
			H21A6	0.60	—	—	
	I_F = 20 mA, V_{CE} = 5 V		H21A4	1.0	—	—	
			H21A5	2.0	—	—	
			H21A6	4.0	—	—	
	I_F = 30 mA, V_{CE} = 5 V		H21A4	1.9	—	—	
			H21A5	3.0	—	—	
			H21A6	5.5	—	—	
Saturation Voltage	I_F = 20 mA, I_C = 1.8 mA	$V_{CE(SAT)}$	H21A5/6	—	—	0.40	V
	I_F = 30 mA, I_C = 1.8 mA		H21A4	—	—	0.40	V
Turn-On Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{on}	All	—	8	—	μs
Turn-Off Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{off}	All	—	50	—	μs

FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

H21A4 H21A5 H21A6

TYPICAL PERFORMANCE CURVES

Figure 1. Output Current vs. Input Current

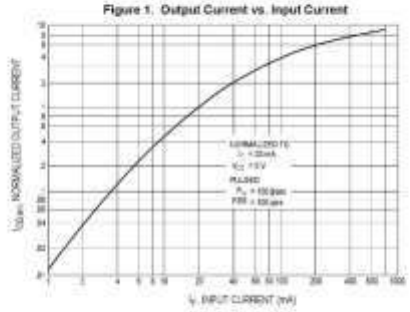


Figure 2. Output Current vs. Temperature

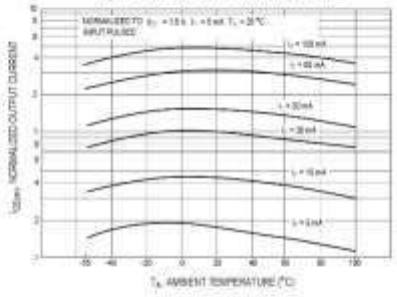


Figure 3. $V_{CE(SAT)}$ vs. Temperature

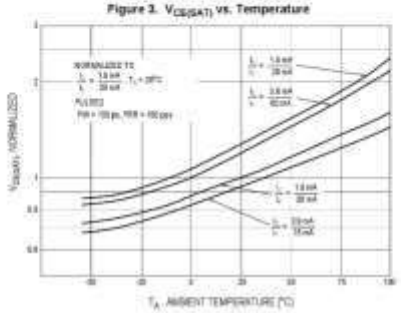


Figure 4. Leakage Current vs. Temperature

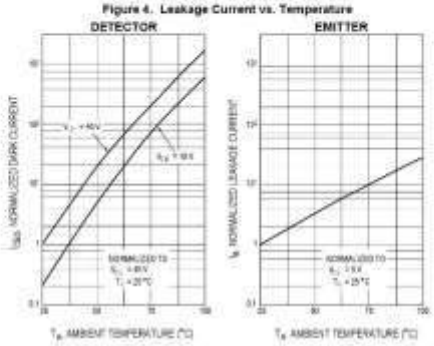


Figure 5. Switching Speed vs. R_L

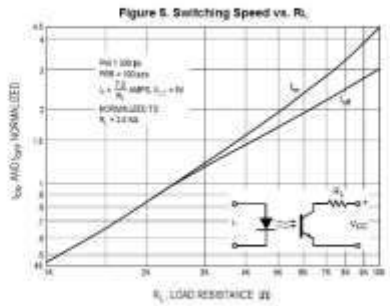
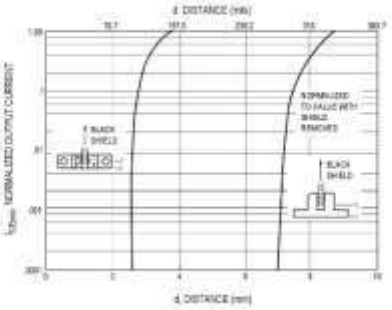


Figure 6. Output Current vs. Distance



FAIRCHILD
SEMICONDUCTOR®

PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

H21A4 H21A5 H21A6

DISCLAIMER
FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY
FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body,or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in labeling, can be reasonably expected to result in a significant injury of the user.

2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.