LAMPIRAN

DATASHEET

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

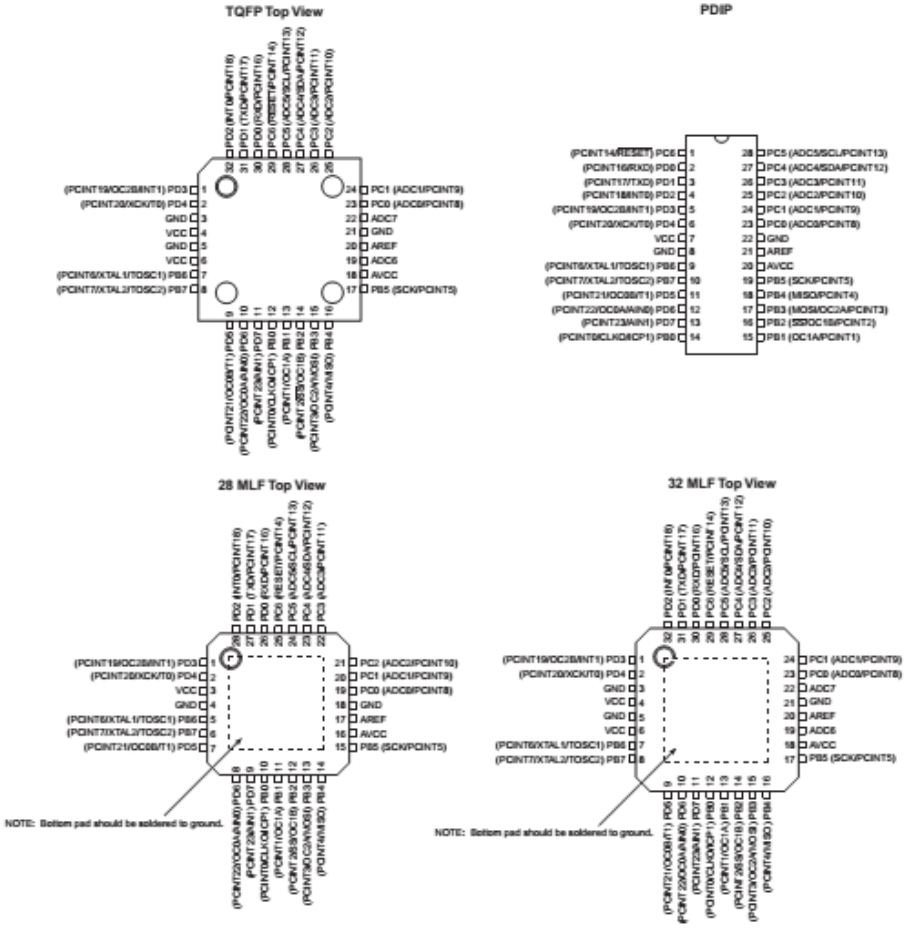
8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P

Rev. 8161D–AVR–10/09

1. Pin Configurations

Figure 1-1. Pinout ATmega48PA/88PA/168PA/328P



19. USART0

19.1 Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

19.2 Overview

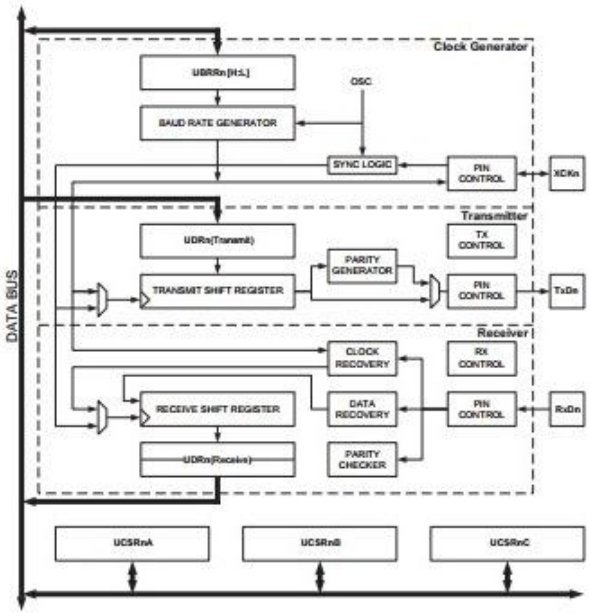
The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

The USART0 can also be used in Master SPI mode, see "USART in SPI Mode" on page 204. The Power Reduction USART bit, PRUSART0, in "Minimizing Power Consumption" on page 42 must be disabled by writing a logical zero to it.

A simplified block diagram of the USART Transmitter is shown in Figure 19-1 on page 177. CPU accessible I/O Registers and I/O pins are shown in bold.

The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (Transfer Clock) pin is only used by synchronous transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register and a two level receive buffer (UDRn). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.

Figure 19-1. USART Block Diagram[1]



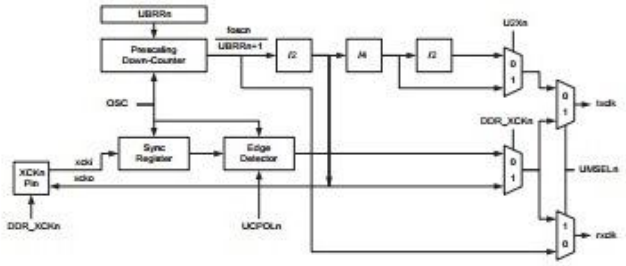
Note: 1. Refer to Figure 1-1 on page 2 and Table 13-9 on page 88 for USART0 pin placement.

19.3 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The UMSELn bit in USART Control and Status Register C (UCSRnC) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2Xn found in the UCSRnA Register. When using synchronous mode (UMSELn = 1), the Data Direction Register for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCKn pin is only active when using synchronous mode.

Figure 19-2 shows a block diagram of the clock generation logic.

Figure 19-2. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (Internal Signal).

rxclk Receiver base clock (Internal Signal).

xcki Input from XCK pin (internal Signal). Used for synchronous slave operation.

xcko Clock output to XCK pin (Internal Signal). Used for synchronous master operation.

fosc XTAL pin frequency (System Clock).

19.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 19-2.

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRRn+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSELn, U2Xn and DDR_XCKn bits.

Table 19-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 19-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{osc}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{osc}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{osc}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System Oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL Registers, (0-4095)

Some examples of UBRRn values for some system clock frequencies are found in Table 19-9 (see page 200).

19.3.2 Double Speed Operation (U2Xn)

The transfer rate can be doubled by setting the U2Xn bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

ITead Studio
Make Innovation easier

Tech Support: info@iteadstudio.com

HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

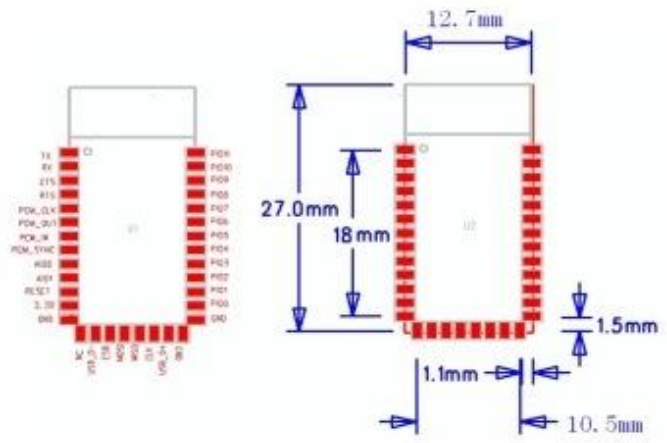
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

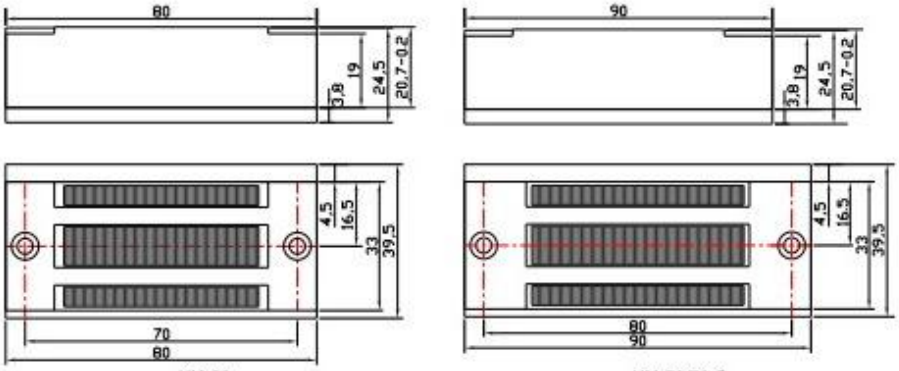
YLI ELECTRONIC

Magnetic Lock Wiring Instructions

Parameters

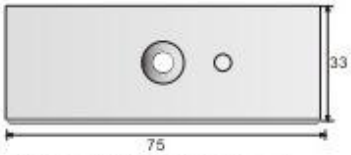
Model	Size (unit:mm)	Magnetic Plate (unit:mm)	Voltage	Current	Holding Force	Door
YM-60	80Lx39.5Wx24.5H	80Lx33Wx19H	12VDC	12V/120mA	60kg(120Lbs)	Single Door
YM-70	90Lx39.5Wx24.5H	90Lx38.5Wx24.5H	12/24VDC	12V/240mA 24V/120mA	70kg(140Lbs)	Single Door
YM-70-S	90Lx39.5Wx24.5H	90Lx38.5Wx24.5H	12/24VDC	12V/240mA 24V/120mA	70kg(140Lbs)	Single Door

Dimension



YM-60

YM-70/70-S



YM-60/YM-70/70-S Armature Plate

Caution:

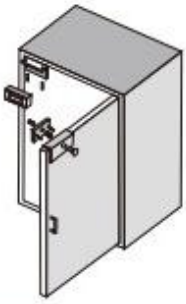
A. The screw of armature plate should not be fixed too tight. Proper elasticity should be guaranteed for the rubber ring so that the armature plate can adjust itself to the appropriate position.

B. Please keep the surface of the lock clean, or the force will be reduced because of the dust, glue or scotch tape on it.

Installation

Demonstration of Standard Installation

Demonstration of Z Bracket Installation



Demonstration of UL Bracket Installation

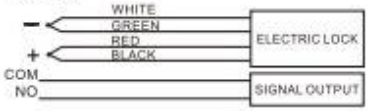
Wiring Diagram

YM-60

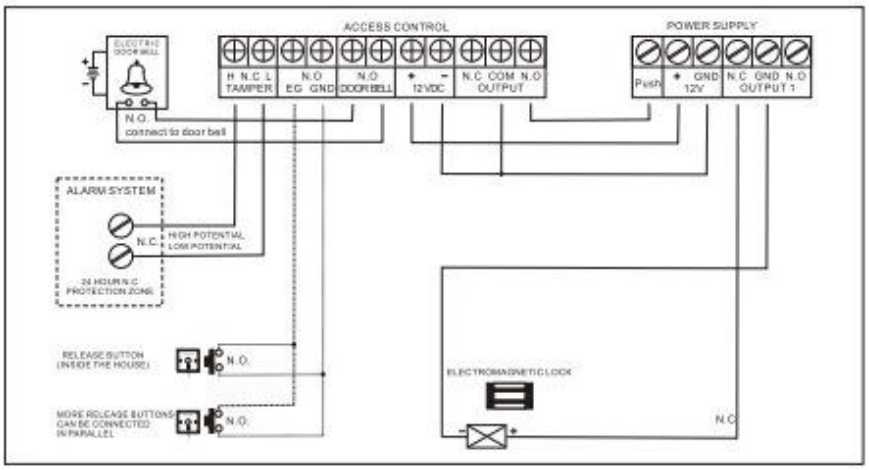
POWER 12VDC 120mA

YM-70/YM-70-S

12V DC



24V DC



www.yli.cn We create security

1. Introduction

This document describes SIM800L hardware interface in great detail.
This document can help user to quickly understand SIM800L interface specifications, electrical and mechanical details. With the help of this document and other SIM800L application notes, user guide, users can use SIM800L to design various applications quickly.

2. SIM800L Overview

SIM800L is a quad-band GSM/GPRS module, that works on frequencies GSM850MHz, EGSM900MHz, DCS1800MHz and PCS1900MHz. SIM800L features GPRS multi-slot class 12/ class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

With a tiny configuration of 15.8*17.8*2.4mm, SIM800L can meet almost all the space requirements in user applications, such as smart phone, PDA and other mobile devices.

SIM800L has 88pin pads of LGA packaging, and provides all hardware interfaces between the module and customers' boards.

- Support 5*5*2 keypads
- One full modem serial port, user can configure two serial ports
- One USB, the USB interfaces can debug, download software
- Audio channel which includes two microphone input; a receiver output and a speaker output
- Programmable general purpose input and output.
- A SIM card interface
- Support FM
- Support one PWM

SIM800L is designed with power saving technique so that the current consumption is as low as 0.7mA in sleep mode.

2.1. SIM800L Key Features

Table 1: SIM800L key features

Feature	Implementation
Power supply	3.4V ~4.4V
Power saving	typical power consumption in sleep mode is 0.7mA (AT+CFUN=0)
Frequency bands	• Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM800L can search the 4 frequency bands automatically. The frequency bands can also be set by AT command "AT+CBAND". For details, please refer to document [1]. • Compliant to GSM Phase 2/2+
Transmitting power	• Class 4 (2W) at GSM 850 and EGSM 900 • Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	• GPRS multi-slot class 12 (default) • GPRS multi-slot class 1~12 (option)
Temperature range	• Normal operation: -40°C ~ +85°C

	● Storage temperature -45°C ~ +90°C
Data GPRS	● GPRS data downlink transfer: max. 85.6 kbps ● GPRS data uplink transfer: max. 85.6 kbps ● Coding scheme: CS-1, CS-2, CS-3 and CS-4 ● PAP protocol for PPP connect ● Integrate the TCP/IP protocol. ● Support Packet Broadcast Control Channel (PBCCH) ● CSD transmission rates：2.4，4.8，9.6，14.4 kbps
CSD	● Support CSD transmission
USSD	● Unstructured Supplementary Services Data (USSD) support
SMS	● MT, MO, CB, Text and PDU mode ● SMS storage: SIM card
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	Speech codec modes: ● Half Rate (ETS 06.20) ● Full Rate (ETS 06.10) ● Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80) ● Adaptive multi rate (AMR) ● Echo Cancellation ● Noise Suppression
Serial port and debug port	**Serial port:** ● Full modem interface with status and control lines, unbalanced, asynchronous. ● 1200bps to 115200bps. ● Can be used for AT commands or data stream. ● Support RTS/CTS hardware handshake and software ON/OFF flow control. ● Multiplex ability according to GSM 07.10 Multiplexer Protocol. ● Autobauding supports baud rate from 1200 bps to 57600bps. ● upgrading firmware **Debug port:** ● USB_DM and USB_DP ● Can be used for debugging and upgrading firmware.
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC.
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Timing functions	Use AT command set
Physical characteristics	Size:15.8*17.8*2.4mm Weight:1.35g
Firmware upgrade	Main serial port or USB port.

2.3. Functional Diagram

The following figure shows a functional diagram of SIM800L:

- GSM baseband
- GSM RF
- Antenna interface
- Other interface

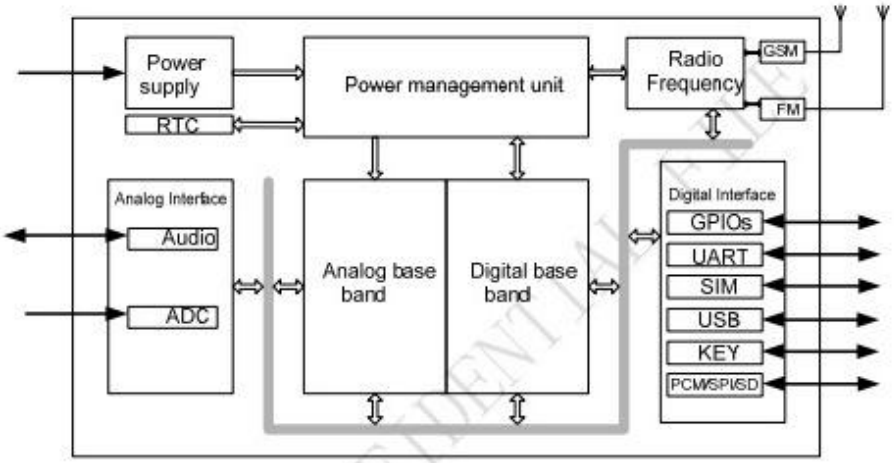


Figure 1: SIM800L functional diagram

3. Package Information

3.1. Pin out Diagram

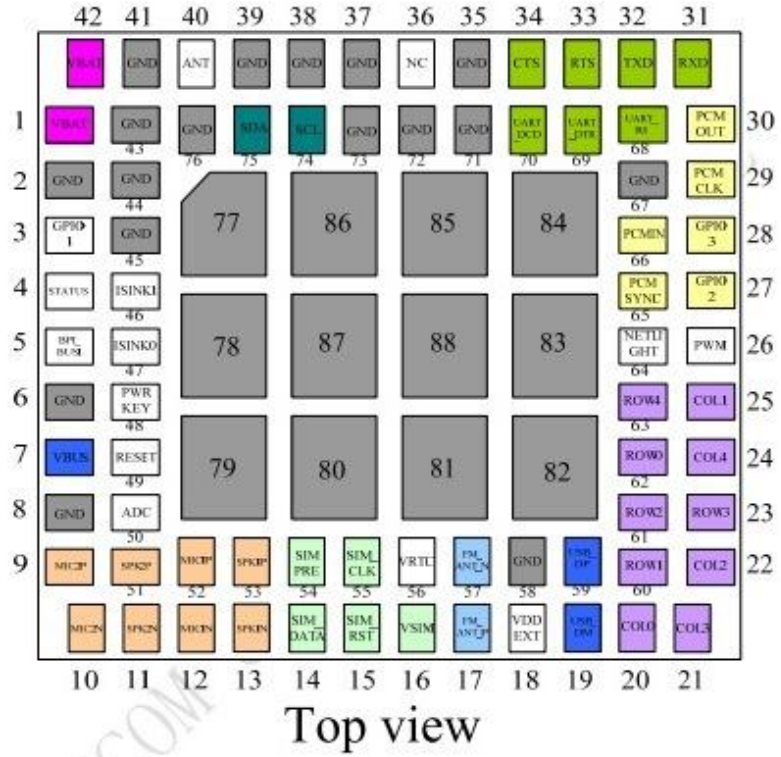


Figure 2: SIM800L pin out diagram (Top view)

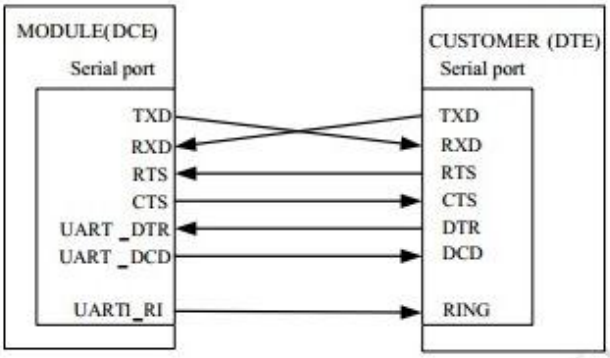


Figure 19: Connection of the serial interfaces

If the voltage of UART is 3.3V, the following reference circuits are recommended. If the voltage is 3.0V, please change the resistors in the following figure from 5.6K to 14K.

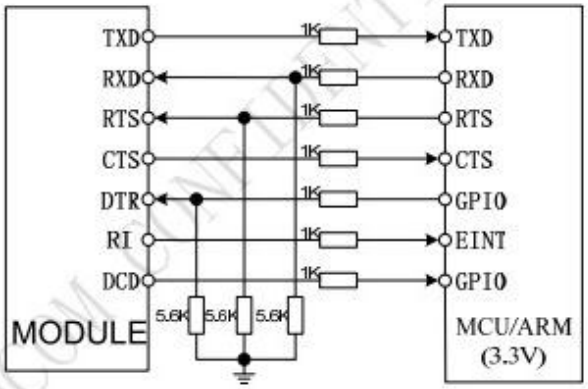


Figure 20: Resistor matching circuit

If the voltage of UART is 3V or3.3V, the following reference circuits are recommended:

5.3 Recommended PCB Layout

Based on above principles, recommended layout is shown in the following illustration.

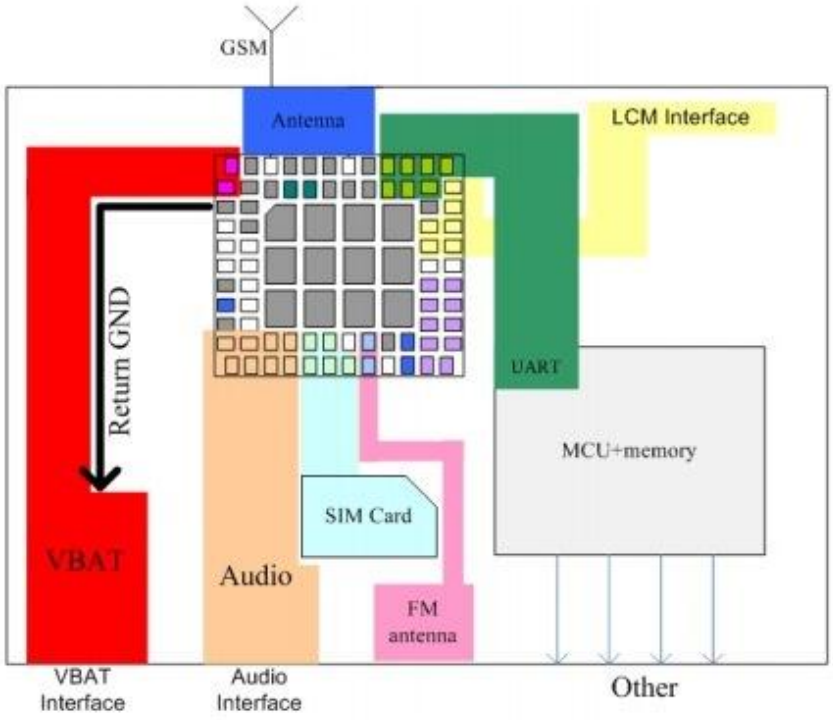


Figure 52: Recommended PCB layout

SOURCE PROGRAM ARDUINO

```
#include <SoftwareSerial.h>          // import the serial library

SoftwareSerial bluetooth(9, 8);      // RX, TX
SoftwareSerial SIM800L(2, 3);        // RX | TX SIM800L

const int switchPin = 10;           // switch detektor status pintu
const int Buzz = 5;                  // Pin kontrol Buzzer
const int doorPin = 7;               // Pin kontrol Pintu

char DATA;                      // variable penyimpan data bluetooth

int counterSMS=0;               // hitung berjalan pengiriman SMS
int maxSMS= 3;                  // maksimum SMS dikirim 3 kali
int loop_SMS3x= 0;              // perulangan kirim SMS


void setup() {
Serial.begin(9600);

bluetooth.begin(38400);
bluetooth.println("Bluetooth ON, Tekan '1' untuk MEMBUKA atau '0'
untuk MENUTUP :.. ");

pinMode(doorPin,OUTPUT);
digitalWrite(doorPin, LOW);

SIM800L.begin(9600);
tesSMS();                       // Kirim TES SMS sebagai penanda
sistem berjalan

pinMode(switchPin, INPUT_PULLUP); // inisialisasi pin 5 sebagai
OUTPUT buzzer

pinMode(Buzz,OUTPUT);
beep(50);                      // bunyikan buzzer 50 ms
beep(50);                     // bunyikan buzzer 50 ms
beep(50);                    // bunyikan buzzer 50 ms
}

void loop() {
bluetooth.listen();     // bluetooth mendengarkan komunikasi data
masuk dari smartphone
while (bluetooth.available() > 0){   // Jika ada bluetooth
terkoneksi
DATA = bluetooth.read();      // Baca data yang masuk, tampung
pada variable DATA
     if(DATA=='0'){                  // Jika karakter "0" diterima
       digitalWrite(doorPin, LOW); // Kunci pintu brankas
       delay(50);
       beep(200);              // Bunyikan buzzer sebagai penanda
       bluetooth.println("BRANKAS DITUTUP! ");
```

```
      }
      if (DATA=='1'){               // Jika karakter "1" diterima
      digitalWrite(doorPin, HIGH);  // Buka pintu brankas
      delay(50);
      beep(200);              // Bunyikan buzzer sebagai penanda
      bluetooth.println("BRANKAS DIBUKA! ");
  }
}


/*
Deteksi Pembobolan !!!
Jika terdeteksi pembobolan pada pintu brankas saat dalam kondisi
terkunci, dan limit switch sebagai detector pembobolan terbaca
HIGH, maka masuk percabangan while yang memanggil fungsi
kirimSMS(); untuk mengirim SMS 3x pada nomor pemilik brankas
*/

      if ((digitalRead(switchPin) == HIGH) &&
(digitalRead(doorPin) == LOW)){

           while (!loop_SMS3x){
             for (int counterSMS=0; counterSMS<maxSMS;
counterSMS++){
             kirimSMS();            // Kirim SMS PERINGATAN
             delay(1000);
             beep(1000);
             bluetooth.println("AWAS!!! PINTU DIBOBOL!!!");
// print status "AWAS!!! PINTU DIBOBOL!!!"
             }
             loop_SMS3x=1;
            break;
           }
     }
     else if ((digitalRead(switchPin == LOW)) &&
(digitalRead(doorPin == LOW))){
       delay(100);
       loop_SMS3x=0;
       }
}


/*
Fungsi untuk menyalakan buzzer
*/
void beep(unsigned char delayms){
  analogWrite(Buzz, 20);

  delay(delayms);
  analogWrite(Buzz, 0);
  delay(delayms);
}
```

```
/*
Fungsi mengirimkan TES SMS
*/
void tesSMS () {
  delay(100);
  String PesanSMS = ("TES KONEKSI SIM800L");

//  Serial.println("Menyiapkan Pesan Text");
  SIM800L.write("AT+CMGF=1\r\n");
  delay(500);

//  Serial.println("Set SMS ke Nomor Tujuan");
  SIM800L.write("AT+CMGS=\"085641718680\"\r\n");
  delay(500);

//  Serial.println("Mengirim pesan SMS");
  SIM800L.print(PesanSMS);
  Serial.println(PesanSMS);

//  Serial.println("Mengakhiri SMS");
  SIM800L.write((char)26);
  delay(500);

//  Serial.println("SMS Selesai Dikirim!");
  delay(500);
}

/*
Fungsi mengirimkan SMS WARNING
*/

void kirimSMS () {
  delay(100);

  String PesanSMS = ("AWAS!!! TERJADI PEMBOBOLAN!!!");

//  Serial.println("Menyiapkan Pesan Text");
  SIM800L.write("AT+CMGF=1\r\n");
  delay(500);

//  Serial.println("Set SMS ke Nomor Tujuan");
  SIM800L.write("AT+CMGS=\"085641718680\"\r\n");
  delay(500);

//  Serial.println("Mengirim pesan SMS");
  SIM800L.print(PesanSMS);
  Serial.println(PesanSMS);

//  Serial.println("Mengakhiri SMS");
  SIM800L.write((char)26);
  delay(500);

//  Serial.println("SMS Selesai Dikirim!");
  delay(500);
}
```

SOURCE ANDROID E-BRANKAS

BLUETOOTH

```
package com.led.led;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;


public class Bluetooth extends AppCompatActivity {
    public static String EXTRA_ADDRESS = "device_address";
    //widgets
    Button btnPaired;
    ListView devicelist;
    //Bluetooth
    private BluetoothAdapter myBluetooth = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth);

        //Calling widgets
        btnPaired = (Button) findViewById(R.id.button);
        devicelist = (ListView) findViewById(R.id.listView);

        //if the device has bluetooth
        myBluetooth = BluetoothAdapter.getDefaultAdapter();

        if (myBluetooth == null) {
            //Show a mensag. that the device has no
bluetooth adapter
            Toast.makeText(getApplicationContext(),
"Bluetooth Device Not Available", Toast.LENGTH_LONG).show();

            //finish apk
```

```
            finish();
        } else if (!myBluetooth.isEnabled()) {
            //Ask to the user turn the bluetooth on
            Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnBTon, 1);
        }

        btnPaired.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pairedDevicesList();
            }
        });

    }

    private void pairedDevicesList() {
        pairedDevices = myBluetooth.getBondedDevices();
        ArrayList list = new ArrayList();

        if (pairedDevices.size() > 0) {
            for (BluetoothDevice bt : pairedDevices) {
                list.add(bt.getName() + "\n" +
bt.getAddress()); //Get the device's name and the address
            }
        } else {
            Toast.makeText(getApplicationContext(), "TIDAK
DITEMUKAN SAMBUNGAN BLUETOOTH", Toast.LENGTH_LONG).show();
        }

        final ArrayAdapter adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, list);
        devicelist.setAdapter(adapter);

devicelist.setOnItemClickListener(myListClickListener);
//Method called when the device from the list is clicked

    }
    private Set<BluetoothDevice> pairedDevices;
    private AdapterView.OnItemClickListener
myListClickListener = new AdapterView.OnItemClickListener()
{
        public void onItemClick(AdapterView<?> av, View v,
int arg2, long arg3) {
            // Get the device MAC address, the last 17 chars
in the View
            String info = ((TextView)
v).getText().toString();
            String address = info.substring(info.length() -
17);

            // Make an intent to start next activity.
            Intent i = new Intent(Bluetooth.this,
PanelPintu.class);
```

```
                //Change the activity.
                i.putExtra(EXTRA_ADDRESS, address); //this will
        be received at PanelPintu (class) Activity
                // Bluetooth.this.finish();
                startActivity(i);
            }
        };
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action
        bar if it is present.
            getMenuInflater().inflate(R.menu.menu_device_list,
        menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action
        bar will
            // automatically handle clicks on the Home/Up
        button, so long
            // as you specify a parent activity in
        AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                Intent i = new Intent(Bluetooth.this,
        PasswordBaru.class);
                Bluetooth.this.finish();
                startActivity(i);
            }

            return super.onOptionsItemSelected(item);
        }
        }
```

DATA BASE HELPER

```
package com.led.led;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;


public class DataBaseHelper extends SQLiteOpenHelper {
    public DataBaseHelper(Context context, String name,
CursorFactory factory, int version) {
        super(context, name, factory, version);
```

```
    }

    @Override
    public void onCreate(SQLiteDatabase _db) {
        _db.execSQL(LoginDataBaseAdapter.DATABASE_CREATE);
        _db.execSQL(LoginDataBaseAdapter.FIRST_ROW);
    }

    @Override
    public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int
_newVersion) {
        // Log the version upgrade.
        Log.w("TaskDBAdapter", "Upgrading from version " +
_oldVersion + " to " + _newVersion + ", which will destroy all old
data");
        _db.execSQL("DROP TABLE IF EXISTS " + "TEMPLATE");
        onCreate(_db);
    }
}
```

KONFIRMASI PASSWORD

```
package com.led.led;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.Toast;


public class KonfirmasiPassword extends Activity {

    LoginDataBaseAdapter loginDataBaseAdapter;
    EditText password, repassword, securityhint;
    Button register, cancel;
    CheckBox check;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.konfirmasi_password);
```

```
        loginDataBaseAdapter = new LoginDataBaseAdapter(this);
        loginDataBaseAdapter = loginDataBaseAdapter.open();
        password = (EditText) findViewById(R.id.password_edt);
        repassword = (EditText) findViewById(R.id.repassword_edt);
        securityhint = (EditText)
findViewById(R.id.securityhint_edt);
        register = (Button) findViewById(R.id.register_btn);
        cancel = (Button) findViewById(R.id.cancel_btn);
        check = (CheckBox) findViewById(R.id.checkBox1);

        check.setOnCheckedChangeListener(new
OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
                // TODO Auto-generated method stub

                if (!isChecked) {

password.setTransformationMethod(PasswordTransformationMethod.getI
nstance());
                } else {

password.setTransformationMethod(HideReturnsTransformationMethod.g
etInstance());
                }
            }
        });


        register.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                String Pass = password.getText().toString();
                String Secu = securityhint.getText().toString();
                String Repass = repassword.getText().toString();

                if (Pass.equals("") || Repass.equals("") ||
Secu.equals("")) {
                    Toast.makeText(getApplicationContext(),
"Masukkan Semua Input", Toast.LENGTH_LONG).show();
                    return;
                }

                if (!Pass.equals(Repass)) {
                    Toast.makeText(getApplicationContext(),
"Password tidak sesuai", Toast.LENGTH_LONG).show();
                    return;
                } else {
                    // Save the Data in Database
                    loginDataBaseAdapter.insertEntry(Pass, Repass,
```

```
Secu);

                    // reg_btn.setVisibility(View.GONE);
                    Toast.makeText(getApplicationContext(), "Akun
baru berhasil dibuat", Toast.LENGTH_LONG).show();
                    Log.d("PASSWORD", Pass);
                    Log.d("REPASSWORD", Repass);
                    Log.d("SECURITYHINT", Secu);
                    Intent i = new Intent(KonfirmasiPassword.this,
Masuk.class);
                    KonfirmasiPassword.this.finish();
                    startActivity(i);

                }
            }
        });
        cancel.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent ii = new Intent(KonfirmasiPassword.this,
Masuk.class);
                KonfirmasiPassword.this.finish();
                startActivity(ii);
            }
        });

    }

}
```

LOGIN DATA BASE ADAPTER

```
package com.led.led;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

import java.util.HashMap;


public class LoginDataBaseAdapter {

    public static final int NAME_COLUMN = 3;
    static final String DATABASE_NAME = "login.db";
    static final int DATABASE_VERSION = 1;
    static final String DATABASE_CREATE = "create table " +
"LOGIN" +
            "( " + "ID integer primary key autoincrement," +
"PASSWORD text," + "REPASSWORD text," + "SECURITYHINT text) ";
```

```
    static final String FIRST_ROW = "INSERT INTO \"LOGIN\"
(\"ID\",\"PASSWORD\",\"REPASSWORD\",\"SECURITYHINT\") VALUES
(\"1\",\"123456\",\"123456\",\"type: 123456\")";
    private final Context context;
    public SQLiteDatabase db;
    private DataBaseHelper dbHelper;

    public LoginDataBaseAdapter(Context _context) {
        context = _context;
        dbHelper = new DataBaseHelper(context, DATABASE_NAME,
null, DATABASE_VERSION);
    }

    public LoginDataBaseAdapter open() throws SQLException {
        db = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        db.close();
    }

    public SQLiteDatabase getDatabaseInstance() {
        return db;
    }

    public void insertEntry(String password, String repassword,
String securityhint) {
        ContentValues newValues = new ContentValues();
        newValues.put("PASSWORD", password);
        newValues.put("REPASSWORD", repassword);
        newValues.put("SECURITYHINT", securityhint);

        db.insert("LOGIN", null, newValues);

    }

    public int deleteEntry(String password) {
        String where = "PASSWORD=?";
        int numberOFEntriesDeleted = db.delete("LOGIN", where, new
String[]{password});
        return numberOFEntriesDeleted;
    }

    public String getSingleEntry(String password) {
        Cursor cursor = db.query("LOGIN", null, " PASSWORD=?", new
String[]{password}, null, null, null);
        if (cursor.getCount() < 1) // UserName Not Exist
        {
            cursor.close();
            return "NOT EXIST";
        }
        cursor.moveToFirst();
        String repassword =
cursor.getString(cursor.getColumnIndex("REPASSWORD"));
        cursor.close();
```

```
        return repassword;
    }

    public String getAllTags(String a) {
        Cursor c = db.rawQuery("SELECT * FROM " + "LOGIN" + "
where SECURITYHINT = '" + a + "'", null);
        String str = null;
        if (c.moveToFirst()) {
            do {
                str = c.getString(c.getColumnIndex("PASSWORD"));
            } while (c.moveToNext());
        }
        return str;
    }

    public String getsemuatag(String a) {
        Cursor c = db.rawQuery("SELECT * FROM " + "LOGIN" + "
where REPASSWORD = '" + a + "'", null);
        String str = null;
        if (c.moveToFirst()) {
            do {
                str = c.getString(c.getColumnIndex("PASSWORD"));
            } while (c.moveToNext());
        }
        return str;
    }


    public void updateEntry(String password, String repassword,
String securityhint) {
        ContentValues updatedValues = new ContentValues();
        updatedValues.put("PASSWORD", password);
        updatedValues.put("REPASSWORD", repassword);
        updatedValues.put("SECURITYHINT", securityhint);

        String where = "PASSWORD=?";
        db.update("LOGIN", updatedValues, where, new
String[]{password});

    }


    public HashMap<String, String> getAnimalInfo(String id) {
        HashMap<String, String> wordList = new HashMap<String,
String>();
        String selectQuery = "SELECT * FROM LOGIN where
SECURITYHINT='" + id + "'";
        Cursor cursor = db.rawQuery(selectQuery, null);
        if (cursor.moveToFirst()) {
            do {
                wordList.put("PASSWORD", cursor.getString(1));
            } while (cursor.moveToNext());
        }
        return wordList;
    }
}
```

MASUK

```
package com.led.led;


import android.app.Activity;
import android.app.Dialog;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class Masuk extends Activity {

    LoginDataBaseAdapter loginDataBaseAdapter;
    Button login;
    EditText enterpassword;
    TextView forgetpass;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.masuk);

        login = (Button) findViewById(R.id.login_btn);

        enterpassword = (EditText)
findViewById(R.id.password_edt);
        forgetpass = (TextView) findViewById(R.id.textView2);

        loginDataBaseAdapter = new
LoginDataBaseAdapter(getApplicationContext());
        loginDataBaseAdapter.open();



        login.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String Password =
enterpassword.getText().toString();

                String storedPassword =
loginDataBaseAdapter.getSingleEntry(Password);

                if (Password.equals(storedPassword)) {
```

```
                    Toast.makeText(Masuk.this, "Status: Login
Berhasil", Toast.LENGTH_LONG).show();
                    Intent ii = new Intent(Masuk.this,
Bluetooth.class);
                    Masuk.this.finish();
                    startActivity(ii);
                } else if (Password.equals("")) {
                    Toast.makeText(Masuk.this, "Masukkan kata
sandi Anda", Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(Masuk.this, "Kata sandi Anda
salah", Toast.LENGTH_LONG).show();
                }
            }
        });

        forgetpass.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                final Dialog dialog = new Dialog(Masuk.this);
                dialog.getWindow();

dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
                dialog.setContentView(R.layout.forget_search);
                dialog.show();

                final EditText security = (EditText)
dialog.findViewById(R.id.securityhint_edt);
                final TextView getpass = (TextView)
dialog.findViewById(R.id.textView3);

                Button ok = (Button)
dialog.findViewById(R.id.getpassword_btn);
                Button cancel = (Button)
dialog.findViewById(R.id.cancel_btn);

                ok.setOnClickListener(new View.OnClickListener() {

                    public void onClick(View v) {

                        String userName =
security.getText().toString();
                        if (userName.equals("")) {

Toast.makeText(getApplicationContext(), "Masukkan bantuan
pengingat kata sandi Anda", Toast.LENGTH_SHORT).show();
                        } else {
                            String storedPassword =
loginDataBaseAdapter.getAllTags(userName);
                            if (storedPassword == null) {

Toast.makeText(getApplicationContext(), "Masukkan pengingat kata
sandi dengan benar", Toast.LENGTH_SHORT).show();
```

```
                                } else {
                                    Log.d("CARI KATA SANDI",
storedPassword);
                                    getpass.setText(storedPassword);
                                }
                            }
                        }
                    });
                    cancel.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                            // TODO Auto-generated method stub
                            dialog.dismiss();
                        }
                    });

                    dialog.show();
                }
            });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Close The Database
        loginDataBaseAdapter.close();
    }

}
```

PANEL PINTU

```
package com.led.led;

import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.os.Handler;

import java.io.IOException;
import java.util.UUID;
```

```
// created by harislightace

public class PanelPintu extends AppCompatActivity {

    //SPP UUID. Look for it
    //initialisasi
    static final UUID myUUID = UUID.fromString("00001101-0000-
1000-8000-00805F9B34FB");
    Button btnOn, btnOff, btnDis, btn20, btn21, btn30, btn31,
btn40, btn41, btn50, btn51, btn60, btn61;

    String address = null;
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;

    Handler bluetoothIn;
    final int handlerState = 0;
    private StringBuilder recDataString = new StringBuilder();
    private ProgressDialog progress;
    private boolean isBtConnected = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent newint = getIntent();
        address = newint.getStringExtra(Bluetooth.EXTRA_ADDRESS);
//receive the address of the bluetooth device

        byte[] buffer = new byte[256];  // buffer store for the
stream
        int bytes;


        //view of the PanelPintu
        setContentView(R.layout.panel_brankas);

        //call the widgtes
        btnOn = (Button) findViewById(R.id.button2);
        btnOff = (Button) findViewById(R.id.button3);
        btnDis = (Button) findViewById(R.id.button4);
        btn20 = (Button) findViewById(R.id.button5);
        btn21 = (Button) findViewById(R.id.button6);
        btn30 = (Button) findViewById(R.id.button7);
        btn31 = (Button) findViewById(R.id.button8);
        btn40 = (Button) findViewById(R.id.button9);
        btn41 = (Button) findViewById(R.id.button10);
        btn50 = (Button) findViewById(R.id.button11);
        btn51 = (Button) findViewById(R.id.button12);
        btn60 = (Button) findViewById(R.id.button13);
        btn61 = (Button) findViewById(R.id.button14);
        new ConnectBT().execute(); //Call the class to connect

        bluetoothIn = new Handler() {
            public void handleMessage(android.os.Message msg) {
                if (msg.what == handlerState) {
```

```
//if message is what we want
                    String readMessage = (String) msg.obj;
// msg.arg1 = bytes from connect thread
                    recDataString.append(readMessage);
//keep appending to string until ~
                    int endOfLineIndex =
recDataString.indexOf("~");                    // determine the
end-of-line
                    if (endOfLineIndex > 0) {
// make sure there data before ~
                        String dataInPrint =
recDataString.substring(0, endOfLineIndex);    // extract string
                        //txtString.setText("Data Received = " +
dataInPrint);
                        int dataLength = dataInPrint.length();
//get length of data received
                        //txtStringLength.setText("String Length =
" + String.valueOf(dataLength));

                        if (recDataString.charAt(0) == '#')
//if it starts with # we know it is what we are looking for
                        {
                            String sensor0 =
recDataString.substring(1, 6);//get sensor value from string
between indices 1-5
                            String sensor01 = sensor0;
                            if (recDataString.charAt(7) == '+')
                            {String sensor11 =
recDataString.substring(8, 11);}
                            else{String sensor11 =
recDataString.substring(7, 11);}
                            if (recDataString.charAt(15) == '+')
                            {String sensor12 =
recDataString.substring(16, 20);}
                            else{String sensor12 =
recDataString.substring(15, 20);}

                            String sensor1 =
recDataString.substring(7, 11);//same again...
                            String sensor2 =
recDataString.substring(15, 20);
                            String sensor3 =
recDataString.substring(21, 24);



                            float values =
Float.parseFloat(sensor0);
                            if (values ==0){
                                Toast.makeText(getBaseContext(),
"Brankas close", Toast.LENGTH_LONG).show();

btn30.setVisibility(View.INVISIBLE);
                            }

                            else if (values ==1){
```

```
                                Toast.makeText(getBaseContext(),
"Brankas Open", Toast.LENGTH_LONG).show();
                            }

                            else if (values ==2){
                                Toast.makeText(getBaseContext(),
"Brankas Dibobol...!!!!", Toast.LENGTH_LONG).show();
                            btn30.setVisibility(View.VISIBLE);

                            }

                            else if (values >150 && values <200){


                            }

                            else if (values >200 && values <250){

                            }
                            else if (values >250 && values <300){

                            }
                            else if (values >300 && values <350){
                            }
                            else if (values >350 && values <400){

                            }
                            else if (values >400 && values <450){

                            }
                            else if (values >450 && values <500){
                            }
                        }
                        recDataString.delete(0,
recDataString.length());                    //clear all string data
                        // strIncom =" ";
                        dataInPrint = " ";
                    }
                }
            }
        };

        //commands to be sent to bluetooth
        btnOn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnLed();
                Toast.makeText(getBaseContext(), "Brankas OPEN",
Toast.LENGTH_LONG).show();//method to turn on

            }
        });

        btnOff.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                turnOffLed();
                Toast.makeText(getBaseContext(), "Brankas close",
Toast.LENGTH_LONG).show();//method to turn off

            }
        });
        btn20.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnpintu();      //method to turn on
            }
        });

        btn21.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOffpintu();   //method to turn off
            }
        });
        btn30.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnpintu1();      //method to turn on
            }
        });

        btn31.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOffpintu1();   //method to turn off
            }
        });
        btn40.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnpintu2();      //method to turn on
            }
        });

        btn41.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOffpintu2();   //method to turn off
            }
        });
        btn50.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnpintu3();      //method to turn on
            }
        });

        btn51.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                turnOffpintu3();   //method to turn off
            }
        });
        btn60.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOnpintu4();      //method to turn on
            }
        });

        btn61.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                turnOffpintu4();   //method to turn off
            }
        });
        btnDis.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Disconnect(); //close connection
            }
        });


    }

    private void Disconnect() {
        if (btSocket != null) //If the btSocket is busy
        {
            try {
                btSocket.close(); //close connection
            } catch (IOException e) {
                msg("Error");
            }
        }
        finish(); //return to the first layout

    }

    private void turnOffLed() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("0".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnpintu() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TA".toString().getBytes());
            } catch (IOException e) {
```

```
                msg("Error");
            }
        }
    }

    private void turnOffpintu() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TB".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnLed() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("1".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnpintu1() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TC".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOffpintu1() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TD".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnpintu2() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TE".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
```

```
            }
        }
    }

    private void turnOffpintu2() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TG".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnpintu3() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TH".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOffpintu3() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TI".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOnpintu4() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TJ".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
        }
    }

    private void turnOffpintu4() {
        if (btSocket != null) {
            try {

btSocket.getOutputStream().write("TK".toString().getBytes());
            } catch (IOException e) {
                msg("Error");
            }
```

```
        }
    }


    // fast way to call Toast
    private void msg(String s) {
        Toast.makeText(getApplicationContext(), s,
Toast.LENGTH_LONG).show();
    }

    public void run() {
        byte[] buffer = new byte[256];
        int bytes;

        // Keep looping to listen for received messages
        while (true) {
            try {
                bytes = btSocket.getInputStream().read(buffer);
//read bytes from input buffer
                String readMessage = new String(buffer, 0, bytes);
                // Send the obtained bytes to the UI Activity via
handler
                bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();
            } catch (IOException e) {
                break;
            }
        }
    }

    private class ConnectBT extends AsyncTask<Void, Void, Void>
// UI thread
    {
        private boolean ConnectSuccess = true; //if it's here,
it's almost connected

        @Override
        protected void onPreExecute() {
            progress = ProgressDialog.show(PanelPintu.this,
"Connecting...", "Please wait!!!");  //show a progress dialog
        }

        @Override
        protected Void doInBackground(Void... devices) //while the
progress dialog is shown, the connection is done in background
        {
            try {
                if (btSocket == null || !isBtConnected) {
                    myBluetooth =
BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth
device
                    BluetoothDevice dispositivo =
myBluetooth.getRemoteDevice(address);//connects to the device's
address and checks if it's available
                    btSocket =
dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);//cr
```

```
eate a RFCOMM (SPP) connection

BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                    btSocket.connect();//start connection
                }
            } catch (IOException e) {
                ConnectSuccess = false;//if the try failed, you
can check the exception here
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result) //after the
doInBackground, it checks if everything went fine
        {
            super.onPostExecute(result);

            if (!ConnectSuccess) {
                msg("Connection Failed. Is it a SPP Bluetooth? Try
again.");
                finish();
            } else {
                msg("Connected.");
                isBtConnected = true;
            }
            progress.dismiss();
        }
    }
}
```

PASSWORD BARU

```
package com.led.led;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.Toast;


public class PasswordBaru extends Activity {

    LoginDataBaseAdapter loginDataBaseAdapter;
```

```
    EditText password, repassword, securityhint;
    Button changepasword, cancel;
    CheckBox check;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.password_baru);

        loginDataBaseAdapter = new LoginDataBaseAdapter(this);
        loginDataBaseAdapter = loginDataBaseAdapter.open();
        password = (EditText) findViewById(R.id.password_edt);
        repassword = (EditText) findViewById(R.id.repassword_edt);
        securityhint = (EditText)
findViewById(R.id.securityhint_edt);
        changepasword = (Button) findViewById(R.id.chgpasswd_btn);
        cancel = (Button) findViewById(R.id.cancel_btn);
        check = (CheckBox) findViewById(R.id.checkBox1);

        check.setOnCheckedChangeListener(new
OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {
                // TODO Auto-generated method stub

                if (!isChecked) {

password.setTransformationMethod(PasswordTransformationMethod.getI
nstance());
                } else {

password.setTransformationMethod(HideReturnsTransformationMethod.g
etInstance());
                }
            }
        });


        changepasword.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub

                String Pass = password.getText().toString();
                String Secu = securityhint.getText().toString();
                String Repass = repassword.getText().toString();


                if (Pass.equals("") || Repass.equals("") ||
Secu.equals("")) {
                    Toast.makeText(getApplicationContext(),
```

```
"Masukkan Semua Input", Toast.LENGTH_LONG).show();
                    return;
                }

                if (Pass.equals("")) {
                    Toast.makeText(getApplicationContext(),
"Masukkan bantuan pengingat kata sandi Anda",
Toast.LENGTH_SHORT).show();
                } else {
                    String storedPassword =
loginDataBaseAdapter.getsemuatag(Pass);
                    if (storedPassword == null) {
                        Toast.makeText(getApplicationContext(),
"Masukkan pengingat kata sandi dengan benar",
Toast.LENGTH_SHORT).show();
                    } else {
                        loginDataBaseAdapter.deleteEntry(Pass);
                        loginDataBaseAdapter.updateEntry(Pass,
Repass, Secu);
                        Log.d("PASSWORD", Repass);
                        Log.d("REPASSWORD", Repass);
                        Log.d("SECURITYHINT", Secu);
                    Toast.makeText(getApplicationContext(),
"Silakan buat kata sandi baru", Toast.LENGTH_LONG).show();
                    Intent i = new Intent(PasswordBaru.this,
KonfirmasiPassword.class);
                        PasswordBaru.this.finish();
                    startActivity(i);
                    }
                }
            }
        });
        cancel.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent ii = new Intent(PasswordBaru.this,
Masuk.class);
                startActivity(ii);
                PasswordBaru.this.finish();

            }
        });

    }

}
```

SPLASH SCREEN

```
package com.led.led;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.view.WindowManager;

public class splashscreen extends Activity {

    //Set waktu lama splashscreen
    private static int splashInterval = 2000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.splashscreen);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                // TODO Auto-generated method stub
                Intent i = new Intent(splashscreen.this,
Masuk.class);
                startActivity(i);


                //jeda selesai Splashscreen
                this.finish();
            }

            private void finish() {
                // TODO Auto-generated method stub

            }
        }, splashInterval);

    }

    ;

}
```