DAFTAR LAMPIRAN

Lampiran 01. Source IRremoteInfo.

```
#include <IRremote.h>

void setup()
{
  Serial.begin(115200); //You may alter the BAUD rate here as needed
  while (!Serial); //wait until Serial is established - required on some Platforms
  //Runs only once per restart of the Arduino.
  dumpHeader();
  dumpRAWBUF();
  dumpTIMER();
  dumpTimerPin();
  dumpClock();
  dumpPlatform();
  dumpPulseParams();
  dumpSignalParams();
  dumpArduinoIDE();
  dumpDebugMode();
  dumpProtocols();
  dumpFooter();
}

void loop() {
  //nothing to do!
}

void dumpRAWBUF() {
  Serial.print(F("RAWBUF: "));
  Serial.println(RAWBUF);
}

void dumpTIMER() {
  boolean flag = false;
#ifdef IR_USE_TIMER1
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer1")); flag = true;
#endif
#ifdef IR_USE_TIMER2
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer2")); flag = true;
#endif
#ifdef IR_USE_TIMER3
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer3")); flag = true;
#endif
```

```
#ifdef IR_USE_TIMER4
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer4")); flag = true;
#endif
#ifdef IR_USE_TIMER5
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer5")); flag = true;
#endif
#ifdef IR_USE_TIMER4_HS
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer4_HS")); flag = true;
#endif
#ifdef IR_USE_TIMER_CMT
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer_CMT")); flag = true;
#endif
#ifdef IR_USE_TIMER_TPM1
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer_TPM1")); flag = true;
#endif0
#ifdef IR_USE_TIMER_TINY0
  Serial.print(F("Timer defined for use: ")); Serial.println(F("Timer_TINY0")); flag = true;
#endif

  if (!flag) {
    Serial.print(F("Timer Error: ")); Serial.println(F("not defined"));
  }
}

void dumpTimerPin() {
  Serial.print(F("IR Tx Pin: "));
  Serial.println(TIMER_PWM_PIN);
}

void dumpClock() {
  Serial.print(F("MCU Clock: "));
  Serial.println(F_CPU);
}

void dumpPlatform() {
  Serial.print(F("MCU Platform: "));

#if defined(__AVR_ATmega1280__)
  Serial.println(F("Arduino Mega1280"));
#elif  defined(__AVR_ATmega2560__)
  Serial.println(F("Arduino Mega2560"));
#elif defined(__AVR_AT90USB162__)
  Serial.println(F("Teensy 1.0 / AT90USB162"));
  // Teensy 2.0
#elif defined(__AVR_ATmega32U4__)
  Serial.println(F("Arduino Leonardo / Yun / Teensy 1.0 / ATmega32U4"));
```

```
#elif defined(__MK20DX128__) || defined(__MK20DX256__)
  Serial.println(F("Teensy 3.0 / Teensy 3.1 / MK20DX128 / MK20DX256"));
#elif defined(__MKL26Z64__)
  Serial.println(F("Teensy-LC / MKL26Z64"));
#elif defined(__AVR_AT90USB646__)
  Serial.println(F("Teensy++ 1.0 / AT90USB646"));
#elif defined(__AVR_AT90USB1286__)
  Serial.println(F("Teensy++ 2.0 / AT90USB1286"));
#elif defined(__AVR_ATmega644P__) || defined(__AVR_ATmega644__)
  Serial.println(F("Sanguino / ATmega644(P)"));
#elif defined(__AVR_ATmega8P__) || defined(__AVR_ATmega8__)
  Serial.println(F("Atmega8 / ATmega8(P)"));
#elif defined(__AVR_ATtiny84__)
  Serial.println(F("ATtiny84"));
#elif defined(__AVR_ATtiny85__)
  Serial.println(F("ATtiny85"));
#else
  Serial.println(F("ATmega328(P) / (Duemilanove, Diecimila, LilyPad, Mini, Micro, Fio, Nano,
etc)"));
#endif
}

void dumpPulseParams() {
  Serial.print(F("Mark Excess: ")); Serial.print(MARK_EXCESS);; Serial.println(F(" uSecs"));
  Serial.print(F("Microseconds per tick: ")); Serial.print(USECPERTICK);; Serial.println(F("
uSecs"));
  Serial.print(F("Measurement tolerance: ")); Serial.print(TOLERANCE); Serial.println(F("%"));
}

void dumpSignalParams() {
  Serial.print(F("Minimum Gap between IR Signals: ")); Serial.print(_GAP); Serial.println(F("
uSecs"));
}

void dumpDebugMode() {
  Serial.print(F("Debug Mode: "));
#if DEBUG
  Serial.println(F("ON"));
#else
  Serial.println(F("OFF (Normal)"));
#endif
}

void dumpArduinoIDE() {
  Serial.print(F("Arduino IDE version: "));
  Serial.print(ARDUINO / 10000);
```

```
  Serial.write('.');
  Serial.print((ARDUINO % 10000) / 100);
  Serial.write('.');
  Serial.println(ARDUINO % 100);
}

void dumpProtocols() {

  Serial.println(); Serial.print(F("IR PROTOCOLS  ")); Serial.print(F("SEND     "));
Serial.println(F("DECODE"));
  Serial.print(F("============= ")); Serial.print(F("======== "));
Serial.println(F("========"));
  Serial.print(F("RC5:          ")); printSendEnabled(SEND_RC5);
printDecodeEnabled(DECODE_RC6);
  Serial.print(F("RC6:          ")); printSendEnabled(SEND_RC6);
printDecodeEnabled(DECODE_RC5);
  Serial.print(F("NEC:          ")); printSendEnabled(SEND_NEC);
printDecodeEnabled(DECODE_NEC);
  Serial.print(F("SONY:         ")); printSendEnabled(SEND_SONY);
printDecodeEnabled(DECODE_SONY);
  Serial.print(F("PANASONIC:    ")); printSendEnabled(SEND_PANASONIC);
printDecodeEnabled(DECODE_PANASONIC);
  Serial.print(F("JVC:          ")); printSendEnabled(SEND_JVC);
printDecodeEnabled(DECODE_JVC);
  Serial.print(F("SAMSUNG:      ")); printSendEnabled(SEND_SAMSUNG);
printDecodeEnabled(DECODE_SAMSUNG);
  Serial.print(F("WHYNTER:      ")); printSendEnabled(SEND_WHYNTER);
printDecodeEnabled(DECODE_WHYNTER);
  Serial.print(F("AIWA_RC_T501: ")); printSendEnabled(SEND_AIWA_RC_T501);
printDecodeEnabled(DECODE_AIWA_RC_T501);
  Serial.print(F("LG:           ")); printSendEnabled(SEND_LG);
printDecodeEnabled(DECODE_LG);
  Serial.print(F("SANYO:        ")); printSendEnabled(SEND_SANYO);
printDecodeEnabled(DECODE_SANYO);
  Serial.print(F("MITSUBISHI:   ")); printSendEnabled(SEND_MITSUBISHI);
printDecodeEnabled(DECODE_MITSUBISHI);
  Serial.print(F("DISH:         ")); printSendEnabled(SEND_DISH);
printDecodeEnabled(DECODE_DISH);
  Serial.print(F("SHARP:        ")); printSendEnabled(SEND_SHARP);
printDecodeEnabled(DECODE_SHARP);
  Serial.print(F("DENON:        ")); printSendEnabled(SEND_DENON);
printDecodeEnabled(DECODE_DENON);
  Serial.print(F("PRONTO:       ")); printSendEnabled(SEND_PRONTO); Serial.println(F("(Not
Applicable)"));
}
```

```
void printSendEnabled(int flag) {
  if (flag) {
    Serial.print(F("Enabled  "));
  }
  else {
    Serial.print(F("Disabled "));
  }
}

void printDecodeEnabled(int flag) {
  if (flag) {
    Serial.println(F("Enabled"));
  }
  else {
    Serial.println(F("Disabled"));
  }
}

void dumpHeader() {
  Serial.println(F("IRremoteInfo - by AnalysIR (http://www.AnalysIR.com/)"));
  Serial.println(F("            - A helper sketch to assist in troubleshooting issues with the library by reviewing the settings within the IRremote library"));
  Serial.println(F("            - Prints out the important settings within the library, which can be configured to suit the many supported platforms"));
  Serial.println(F("            - When seeking on-line support, please post or upload the output of this sketch, where appropriate"));
  Serial.println();
  Serial.println(F("IRremote Library Settings"));
  Serial.println(F("========================="));
}

void dumpFooter() {
  Serial.println();
  Serial.println(F("Notes: "));
  Serial.println(F("     - Most of the seetings above can be configured in the following files included as part of the library"));
  Serial.println(F("     - IRremteInt.h"));
  Serial.println(F("     - IRremote.h"));
  Serial.println(F("     - You can save SRAM by disabling the Decode or Send features for any protocol (Near the top of IRremoteInt.h)"));
  Serial.println(F("     - Some Timer conflicts, with other libraries, can be easily resolved by configuring a differnt Timer for your platform"));
}
```

Lampiran 02. LG AC Send Data

```
#include <IRremote.h>
#include <Wire.h>
IRsend irsend;
// not used
int RECV_PIN = 11;
IRrecv irrecv (RECV_PIN);
const int AC_TYPE  = 0;
// 0 : TOWER
// 1 : WALL
//
int AC_HEAT = 0;
// 0 : cooling
// 1 : heating
int AC_POWER_ON    = 0;
// 0 : off
// 1 : on
int AC_AIR_ACLEAN  = 0;
// 0 : off
// 1 : on --> power on
int AC_TEMPERATURE = 27;
// temperature : 18 ~ 30
int AC_FLOW        = 1;
// 0 : low
// 1 : mid
// 2 : high
// if AC_TYPE =1, 3 : change
//
const int AC_FLOW_TOWER[3] = {0, 4, 6};
const int AC_FLOW_WALL[4]  = {0, 2, 4, 5};

unsigned long AC_CODE_TO_SEND;
int r = LOW;
int o_r = LOW;
byte a, b;
void ac_send_code(unsigned long code)
{
  Serial.print("code to send : ");
  Serial.print(code, BIN);
  Serial.print(" : ");
  Serial.println(code, HEX);
  irsend.sendLG(code, 28);
}
void ac_activate(int temperature, int air_flow)
{
  int AC_MSBITS1 = 8;
```

```
  int AC_MSBITS2 = 8;
  int AC_MSBITS3 = 0;
  int AC_MSBITS4 ;
  if ( AC_HEAT == 1 ) {
    // heating
    AC_MSBITS4 = 4;
  } else {
    // cooling
    AC_MSBITS4 = 0;
  }
  int AC_MSBITS5 = temperature - 15;
  int AC_MSBITS6 ;

  if ( AC_TYPE == 0) {
    AC_MSBITS6 = AC_FLOW_TOWER[air_flow];
  } else {
    AC_MSBITS6 = AC_FLOW_WALL[air_flow];
  }
  int AC_MSBITS7 = (AC_MSBITS3 + AC_MSBITS4 + AC_MSBITS5 + AC_MSBITS6) &
B00001111;
  AC_CODE_TO_SEND =  AC_MSBITS1 << 4 ;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS2) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS3) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS4) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS5) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS6) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS7);

  ac_send_code(AC_CODE_TO_SEND);

  AC_POWER_ON = 1;
  AC_TEMPERATURE = temperature;
  AC_FLOW = air_flow;
}
void ac_change_air_swing(int air_swing)
{
  if ( AC_TYPE == 0) {
    if ( air_swing == 1) {
      AC_CODE_TO_SEND = 0x881316B;
    } else {
      AC_CODE_TO_SEND = 0x881317C;
    }
  } else {
    if ( air_swing == 1) {
      AC_CODE_TO_SEND = 0x8813149;
    } else {
```

```
    AC_CODE_TO_SEND = 0x881315A;
   }
  }

  ac_send_code(AC_CODE_TO_SEND);
}

void ac_power_down()
{
  AC_CODE_TO_SEND = 0x88C0051;

  ac_send_code(AC_CODE_TO_SEND);

  AC_POWER_ON = 0;
}

void ac_air_clean(int air_clean)
{
  if ( air_clean == 1) {
    AC_CODE_TO_SEND = 0x88C000C;
  } else {
    AC_CODE_TO_SEND = 0x88C0084;
  }

  ac_send_code(AC_CODE_TO_SEND);
  AC_AIR_ACLEAN = air_clean;
}
void setup()
{
  Serial.begin(38400);
  delay(1000);
  Wire.begin(7);
  Wire.onReceive(receiveEvent);
  Serial.println("  - - - T E S T - - -   ");

  /* test
    ac_activate(25, 1);
    delay(5000);
    ac_activate(27, 2);
    delay(5000);

  */
}

void loop()
{
```

```
ac_activate(25, 1);
delay(5000);
ac_activate(27, 0);
delay(5000);
if ( r != o_r) {
  /*
  # a : mode or temp    b : air_flow, temp, swing, clean, cooling/heating
  # 18 ~ 30 : temp      0 ~ 2 : flow // on
  # 0 : off             0
  # 1 : on              0
  # 2 : air_swing       0 or 1
  # 3 : air_clean       0 or 1
  # 4 : air_flow        0 ~ 2 : flow
  # 5 : temp           18 ~ 30
  # + : temp + 1
  # - : temp - 1
  # m : change cooling to air clean, air clean to cooling
  */
  Serial.print("a : ");
  Serial.print(a);
  Serial.print("  b : ");
  Serial.println(b);

  switch (a) {
    case 0: // off
      ac_power_down();
      break;
    case 1: // on
      ac_activate(AC_TEMPERATURE, AC_FLOW);
      break;
    case 2:
      if ( b == 0 | b == 1 ) {
        ac_change_air_swing(b);
      }
      break;
    case 3: // 1  : clean on, power on
      if ( b == 0 | b == 1 ) {
        ac_air_clean(b);
      }
      break;
    case 4:
      if ( 0 <= b && b <= 2  ) {
        ac_activate(AC_TEMPERATURE, b);
      }
      break;
    case 5:
```

```
      if (18 <= b && b <= 30  ) {
        ac_activate(b, AC_FLOW);
      }
      break;
    case '+':
      if ( 18 <= AC_TEMPERATURE && AC_TEMPERATURE <= 29 ) {
        ac_activate((AC_TEMPERATURE + 1), AC_FLOW);
      }
      break;
    case '-':
      if ( 19 <= AC_TEMPERATURE && AC_TEMPERATURE <= 30 ) {
        ac_activate((AC_TEMPERATURE - 1), AC_FLOW);
      }
      break;
    case 'm':
      /*
        if ac is on,  1) turn off, 2) turn on ac_air_clean(1)
        if ac is off, 1) turn on,  2) turn off ac_air_clean(0)
      */
      if ( AC_POWER_ON == 1 ) {
        ac_power_down();
        delay(100);
        ac_air_clean(1);
      } else {
        if ( AC_AIR_ACLEAN == 1) {
          ac_air_clean(0);
          delay(100);
        }
        ac_activate(AC_TEMPERATURE, AC_FLOW);
      }
      break;
    default:
      if ( 18 <= a && a <= 30 ) {
        if ( 0 <= b && b <= 2 ) {
          ac_activate(a, b);
        }
      }
   }
   o_r = r ;
  }
  delay(100);
}
void receiveEvent(int howMany)
{
  a = Wire.read();
  b = Wire.read();
```

```
  r = !r ;
}
```

Lampiran 03. Coding keseluruhan program Arduino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <HCSR04.h>
#include <IRremote.h>


LiquidCrystal_I2C lcd(0x3f,20,4);

UltraSonicDistanceSensor distanceSensor1(12,13);
UltraSonicDistanceSensor distanceSensor2(9,10);// echo,triger

IRsend irsend;

/*
 * IRremote: IRsendDemo - demonstrates sending IR codes with IRsend
 * An IR LED must be connected to Arduino PWM pin 3.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 *
 *
*/

#define off 0x1035C9DA
#define on_16 0xB62439F1
#define on_17 0x579E9CA6
#define on_18 0xA81ACB9B
#define on_19 0x2CFB0AD6
#define on_20 0xBC16E727
#define on_21 0xE4C60FAA
#define on_22 0x72408D8F
#define on_23 0xCA68D726
#define on_24 0xD7521A0D
#define on_25 0x8A271E3A
#define on_26 0x48B35411
#define on_27 0x16CB355A
```

```
#define on_28 0xA5E711AB
#define on_29 0xCE963A2E
#define on_30 0xCEC2288D


#define DETEKSI_ORANG 1
#define KONTROL_TEMPERATUR 2
#define KIRIM_PERINTAH 3
#define PRINT_DATA 4
uint8_t state_machine=DETEKSI_ORANG;

const uint8_t jarak_tembok=30;
const uint8_t toleransi=5;

double sensor_jarak1=(double)jarak_tembok;
double sensor_jarak2=(double)jarak_tembok;
int8_t jumlah_orang=0;
int8_t jumlah_orang_sebelumnya=0;

bool state_backlight=LOW;
unsigned long time_backlight=0;
unsigned long time_send=0;

/**********************************************************IR CONTROLL
Variable**********************************************************/
const int AC_TYPE  = 1;
// 0 : TOWER
// 1 : WALL
//

int AC_HEAT = 0;
// 0 : cooling
// 1 : heating

int AC_POWER_ON    = 0;
// 0 : off
// 1 : on

int AC_AIR_ACLEAN  = 0;
// 0 : off
// 1 : on --> power on

int AC_TEMPERATURE= 27;
// temperature : 18 ~ 30

int AC_FLOW        = 1;
```

```
// 0 : low
// 1 : mid
// 2 : high
// if AC_TYPE =1, 3 : change
//


const int AC_FLOW_TOWER[3] = {0, 4, 6};
const int AC_FLOW_WALL[4]  = {0, 2, 4, 5};

unsigned long AC_CODE_TO_SEND;

int r = LOW;
int o_r = LOW;

byte a, b;

/**********************************************************END IR CONTROLL
Variable**********************************************************/
void ac_send_code(unsigned long code)
{
  Serial.print("code to send : ");
  Serial.print(code, BIN);
  Serial.print(" : ");
  Serial.println(code, HEX);

  irsend.sendLG(code, 28);
}

void ac_activate(int temperature, int air_flow)
{

  int AC_MSBITS1 = 8;
  int AC_MSBITS2 = 8;
  int AC_MSBITS3 = 0;
  int AC_MSBITS4 ;
  if ( AC_HEAT == 1 ) {
    // heating
    AC_MSBITS4 = 4;
  } else {
    // cooling
    AC_MSBITS4 = 0;
  }
  int AC_MSBITS5 = temperature - 15;
  int AC_MSBITS6 ;
```

```
  if ( AC_TYPE == 0) {
    AC_MSBITS6 = AC_FLOW_TOWER[air_flow];
  } else {
    AC_MSBITS6 = AC_FLOW_WALL[air_flow];
  }

  int AC_MSBITS7 = (AC_MSBITS3 + AC_MSBITS4 + AC_MSBITS5 + AC_MSBITS6) &
B00001111;

  AC_CODE_TO_SEND =  AC_MSBITS1 << 4 ;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS2) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS3) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS4) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS5) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS6) << 4;
  AC_CODE_TO_SEND =  (AC_CODE_TO_SEND + AC_MSBITS7);

  ac_send_code(AC_CODE_TO_SEND);

  AC_POWER_ON = 1;
  AC_TEMPERATURE = temperature;
  AC_FLOW = air_flow;
}

void ac_change_air_swing(int air_swing)
{
  if ( AC_TYPE == 0) {
    if ( air_swing == 1) {
      AC_CODE_TO_SEND = 0x881316B;
    } else {
      AC_CODE_TO_SEND = 0x881317C;
    }
  } else {
    if ( air_swing == 1) {
      AC_CODE_TO_SEND = 0x8813149;
    } else {
      AC_CODE_TO_SEND = 0x881315A;
    }
  }

  ac_send_code(AC_CODE_TO_SEND);
}

void ac_power_down()
{
  AC_CODE_TO_SEND = 0x88C0051;
```

```
  ac_send_code(AC_CODE_TO_SEND);

  AC_POWER_ON = 0;
}

void ac_air_clean(int air_clean)
{
  if ( air_clean == 1) {
    AC_CODE_TO_SEND = 0x88C000C;
  } else {
    AC_CODE_TO_SEND = 0x88C0084;
  }

  ac_send_code(AC_CODE_TO_SEND);

  AC_AIR_ACLEAN = air_clean;
}

void sendIRCommand(){
  if(AC_TEMPERATURE==0)ac_power_down();//off
  else{
    if(AC_TEMPERATURE<18) AC_TEMPERATURE=18;
    else if(AC_TEMPERATURE>30) AC_TEMPERATURE=30;
    else ac_activate(AC_TEMPERATURE, AC_FLOW);  //(0N) TEMP, FLOW
  }


//          ac_change_air_swing(b);
//          ac_air_clean(b);
//          ac_activate(AC_TEMPERATURE, b);
//          ac_activate(b, AC_FLOW);
//
//       if ( 18 <= AC_TEMPERATURE && AC_TEMPERATURE <= 29 ) {
//          ac_activate((AC_TEMPERATURE + 1), AC_FLOW);
//
//        if ( 19 <= AC_TEMPERATURE && AC_TEMPERATURE <= 30 ) {
//          ac_activate((AC_TEMPERATURE - 1), AC_FLOW);
//
//        /*
//          if ac is on,  1) turn off, 2) turn on ac_air_clean(1)
//          if ac is off, 1) turn on,  2) turn off ac_air_clean(0)
//        */
//        if ( AC_POWER_ON == 1 ) {
//          ac_power_down();
//          delay(100);
```

```
//          ac_air_clean(1);
//        } else {
//          if ( AC_AIR_ACLEAN == 1) {
//            ac_air_clean(0);
//            delay(100);
//          }
//          ac_activate(AC_TEMPERATURE, AC_FLOW);
//        }
}



/*************************************************************fungsi
ALAT*************************************************************/
void readSensor(){
  sensor_jarak1=distanceSensor1.measureDistanceCm(); delay(10);
  if (sensor_jarak1<jarak_tembok-toleransi){
    unsigned long time_sensor=millis();

    while(millis()-time_sensor<=500){ //menunggu melewati sensor ke 2  //1000
      sensor_jarak2=distanceSensor2.measureDistanceCm();
      if(sensor_jarak2<jarak_tembok-toleransi){
        jumlah_orang++;
        break;
      }
    }
  }

  else {
    sensor_jarak2=distanceSensor2.measureDistanceCm(); delay(10);
    if (sensor_jarak2<jarak_tembok-toleransi){
      unsigned long time_sensor=millis();

      while(millis()-time_sensor<=500){ //menunggu melewati sensor ke 2 //1000
        sensor_jarak1=distanceSensor1.measureDistanceCm();
        if(sensor_jarak1<jarak_tembok-toleransi){
          jumlah_orang--;
          break;
        }
      }
    }
  }


  Serial.print(sensor_jarak1);
  Serial.print('\t');
```

```
  Serial.print(sensor_jarak2);
  Serial.print('\t');
  Serial.println(jumlah_orang);
}

void setTemp(){
  if(jumlah_orang<1) { AC_TEMPERATURE=0;}
  else if(jumlah_orang<=1) { AC_TEMPERATURE=25;}
  else if(jumlah_orang<=2) { AC_TEMPERATURE=24;}
  else if(jumlah_orang<=3) { AC_TEMPERATURE=23;}
  else if(jumlah_orang<=4) { AC_TEMPERATURE=22;}
  else if(jumlah_orang<=5) { AC_TEMPERATURE=21;}
  else if(jumlah_orang<=6) { AC_TEMPERATURE=20;}
  else if(jumlah_orang<=7) { AC_TEMPERATURE=19;}
  else if(jumlah_orang<=8) { AC_TEMPERATURE=18;}
}

void printData(){
  lcd.setCursor(0,1);
  lcd.print("T:");
  if(AC_TEMPERATURE==0) lcd.print("OFF");
  else {
    lcd.print(AC_TEMPERATURE);
    lcd.print("C | ");
  }
  lcd.print(jumlah_orang);
  lcd.print(" Orang  ");
}

void setup()
{
  Serial.begin(9600);
  pinMode(3,OUTPUT);

  lcd.init();

  // Print a message to the LCD.
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Starting. . .");
  delay(500);
  lcd.setCursor(0,0);
  lcd.print("Controller LG AC");
  lcd.noBacklight(); delay(100);
  lcd.backlight();delay(100);
  lcd.noBacklight(); delay(100);
```

```
  lcd.backlight();delay(100);
  lcd.noBacklight(); delay(100);
}

void loop() {
  switch(state_machine)
  {
    case DETEKSI_ORANG:
      readSensor();
      if(millis()-time_send>=2000) state_machine=KIRIM_PERINTAH;
      else if(jumlah_orang != jumlah_orang_sebelumnya)
state_machine=KONTROL_TEMPERATUR;
      else if(state_backlight==HIGH && millis()-time_backlight>=2000) {lcd.noBacklight();
state_backlight=LOW;}
    break;

    case KIRIM_PERINTAH:
      sendIRCommand();
      time_send=millis();
      state_machine=DETEKSI_ORANG;
    break;

    case KONTROL_TEMPERATUR:
      setTemp();
      jumlah_orang_sebelumnya=jumlah_orang;
      state_machine=PRINT_DATA;
    break;

    case PRINT_DATA:
      lcd.backlight();
      printData();
      time_backlight=millis(); state_backlight=HIGH;
      delay(500);
      state_machine=DETEKSI_ORANG;
    break;
  }
  //readSensor();
}
```