LAMPIRAN

Lampiran 1.1 Pengujian 1 Sensor terhadap Objek Manusia dengan Jarak 30-40 cm

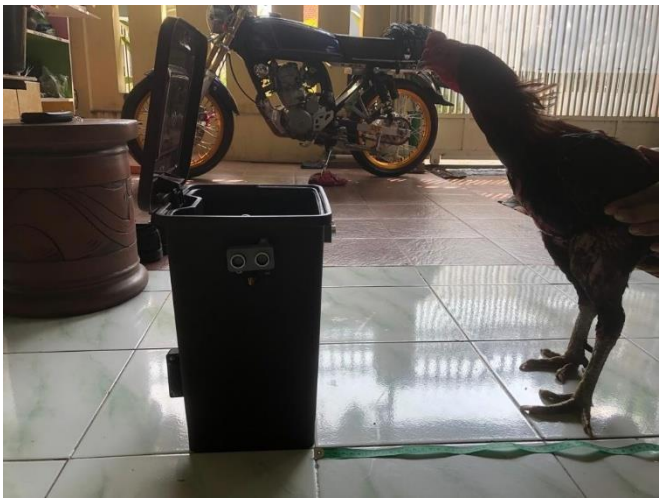
Lampiran 1.2 Pengujian 1 Sensor terhadap Objek Manusia dengan Jarak 40-50cm

Lampiran 1.3 Pengujian 1 Sensor terhadap Objek Manusia dengan Jarak 50-60cm

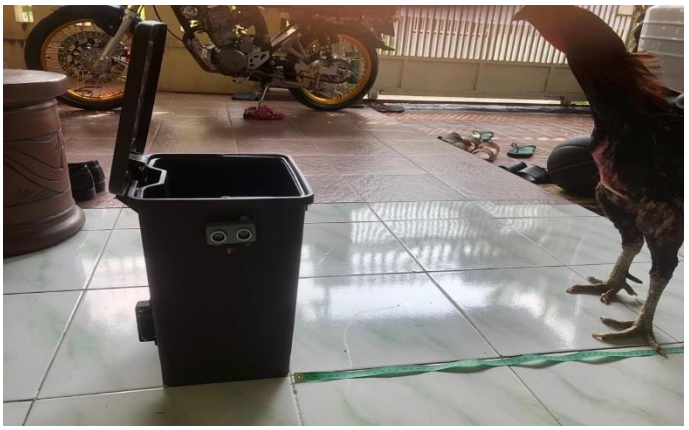
Lampiran 1.4 Pengujian 1 Sensor terhadap Objek Benda dengan Jarak 30-40cm

Lampiran 1.5 Pengujian 1 Sensor terhadap Objek Benda dengan Jarak 40-50cm

Lampiran 1.6 Pengujian 1 Sensor terhadap Objek Benda dengan Jarak 50-60cm



Lampiran 1.7 Pengujian 1 Sensor terhadap Objek Hewan dengan Jarak 30-40cm



Lampiran 1.8 Pengujian 1 Sensor terhadap Objek Hewan dengan Jarak 40-50cm

Lampiran 1.9 Pengujian 1 Sensor terhadap Objek Hewan dengan Jarak 50-60cm

Lampiran 1.10 Pengujian 2 Sensor Secara Bersamaan dengan Jarak 30-40cm

Lampiran 1.11 Pengujian 2 Sensor Secara Bersamaan dengan Jarak 40-50cm

Lampiran 1.12 Pengujian 2 Sensor Secara Bersamaan dengan Jarak 50-60cm

Lampiran 1.13 Pengujian Kombinasi Semua Sensor dengan Jarak 30-40cm

Lampiran 1.14 Pengujian Kombinasi Semua Sensor dengan Jarak 40-50cm

Lampiran 1.15 Pengujian kombinasi semua sensor dengan jarak 50-60cm

Lampiran 1.16 Pengujian diantara Sudut Sensor 1 dan 2

Lampiran 1.17 Pengujian diantara Sudut Sensor 1 dan 3

Program Utama Tes Sensor

Pemrograman sensor dilakukan pada komputer untuk mengetahui fungsi sensordengan jarak pada sensor yang kita inginkan, dengan program sebagai berikut :

```
#include <NewPing.h>
#define TRIGGER_PIN  3     //   pin   trigger
terkoneksi ke arduino#define ECHO_PIN     4
// pinecho terkoneksi pada arduino
#define MAX_DISTANCE 200 // jarak maksimal yang dibatasi oleh
software untukpembacaan sensor
NewPing  sonar(TRIGGER_PIN,  ECHO_PIN,  MAX_DISTANCE);  //
set  up  pin  yangdigunakan di diproses pada library
void setup() {
  Serial.begin(115200);  //  kecepatan  serial(kecepatan  data
untuk melakukanlaporan pada komputer).
}
void loop() {
  delay(100);                              //   waktu   tunggu
sonar  balik  pada  sensor(tdk boleh  lebih  kecil  dari
29ms)satuan ms pembacaan ideal 100ms
Serial.print("jarak : ");
Serial.print(sonar.ping_cm()); // proses pembacaan sensor
Serial.println("cm") ;              // proses waktu tunggu
pelaporan pada komputer}
```

Program Penunjang Untuk Tes Sensor

Pemrograman peenunjang untuk tes sensor ini dilakukan untuk memperlancar dan meningkatkan fungsi sensor , berikut program penunjang tes sensor :

```
#define NewPing_h
#if defined (ARDUINO) && ARDUINO >= 100
#include <Arduino.h>
#else
#include <WProgram.h>
#if defined (PARTICLE)
#include <SparkIntervalTimer.h>
#else
#include <pins_arduino.h>
#endif
#endif
#if defined (__AVR__)
#include <avr/io.h>
#include <avr/interrupt.h>
```

```
#endif // Shouldn't need to change these values unless you have a                specific need to do so.
#define MAX_SENSOR_DISTANCE 500 // Maximum sensor distance can be as high as 500cm, no reason to wait for ping longer than sound takes to travel this distance and back. Default=500
#define US_ROUNDTRIP_CM 57      // Microseconds (uS) it takes sound to travel round-trip 1cm (2cm total), uses integer to save compiled code space. Default=57
#define US_ROUNDTRIP_IN 146     // Microseconds (uS) it takes sound to travel round-trip 1 inch (2 inches total), uses integer to save compiled code space. Defalult=146
#define ONE_PIN_ENABLED true    // Set to "false" to disable one pin mode which saves around 14-26 bytes of binary size. Default=true
#define ROUNDING_ENABLED false  // Set to "true" to enable distance rounding which also adds 64 bytes to binary size. Default=false
#define URM37_ENABLED false     // Set to "true" to enable support for the URM37 sensor in PWM mode. Default=false//  Probably shouldn't change these values unless you really know what you're doing.
#define NO_ECHO 0               // Value returned if there's no ping echo within the specified MAX_SENSOR_DISTANCE or max_cm_distance. Default=0
#define MAX_SENSOR_DELAY 5800   // Maximum uS it takes for sensor to start the ping. Default=5800
#define ECHO_TIMER_FREQ 24      // Frequency to check for a ping echo (every 24uS is about 0.4cm accuracy). Default=24
#define PING_MEDIAN_DELAY 29000 // Microsecond delay between pings in the ping_median method. Default=29000
#if URM37_ENABLED == true
#undef  US_ROUNDTRIP_CM
#undef  US_ROUNDTRIP_IN
#define US_ROUNDTRIP_CM 50  // Every 50uS PWM signal is low indicates 1cm distance. Default=50
#define US_ROUNDTRIP_IN 127 // If 50uS is 1cm, 1 inch would be 127uS (50 x 2.54 = 127). Default=127
#endif// Conversion from uS to distance (round result to nearest cm or inch).
#define  NewPingConvert(echoTime,  conversionFactor) (max(((unsigned int)echoTime + conversionFactor / 2) / conversionFactor, (echoTime ? 1 : 0)))// Detect non-AVR microcontrollers (Teensy 3.x, Arduino DUE, etc.) and don't use port registers or timer interrupts as
```

```
required.
#if (defined (__arm__) && (defined (TEENSYDUINO) ||
defined (PARTICLE)))
#define PING_OVERHEAD 1
#define PING_TIMER_OVERHEAD 1
#define TIMER_ENABLED true
#define DO_BITWISE true
#elif defined (__AVR__)
#define PING_OVERHEAD 5           // Ping overhead in
microseconds (uS). Default=5
#define PING_TIMER_OVERHEAD 13  // Ping timer overhead
in microseconds (uS). Default=13
#define TIMER_ENABLED true
#define DO_BITWISE true
#else
#define PING_OVERHEAD 1
#define PING_TIMER_OVERHEAD 1
#define TIMER_ENABLED false
#define DO_BITWISE false
#endif// Disable the timer interrupts when using
ATmega128,     ATmega4809     and     all     ATtiny
microcontrollers.
#if       defined       (__AVR_ATmega128__)       ||
defined(__AVR_ATmega4809__)         ||         defined
(__AVR_ATtiny24__) || defined (__AVR_ATtiny44__) ||
defined      (__AVR_ATtiny441__)      ||      defined
(__AVR_ATtiny84__) || defined (__AVR_ATtiny841__) ||
defined      (__AVR_ATtiny25__)      ||      defined
(__AVR_ATtiny45__) || defined (__AVR_ATtiny85__) ||
defined      (__AVR_ATtiny261__)      ||      defined
(__AVR_ATtiny461__) || defined (__AVR_ATtiny861__) ||
defined (__AVR_ATtiny43U__)
#undef  TIMER_ENABLED
#define TIMER_ENABLED false
#endif// Define timers when using ATmega8, ATmega16,
ATmega32 and ATmega8535 microcontrollers.
#if    defined    (__AVR_ATmega8__)    ||    defined
(__AVR_ATmega16__) || defined (__AVR_ATmega32__) ||
defined (__AVR_ATmega8535__)
#define OCR2A OCR2
#define TIMSK2 TIMSK
#define OCIE2A OCIE2
#endif
class  NewPing  {public:NewPing(uint8_t  trigger_pin,
uint8_t  echo_pin,  unsigned  int  max_cm_distance  =
MAX_SENSOR_DISTANCE);
unsigned int ping(unsigned int max_cm_distance = 0);
```

```
unsigned long ping_cm(unsigned int max_cm_distance =
0);
unsigned long ping_in(unsigned int max_cm_distance =
0);
unsigned long ping_median(uint8_t it = 5, unsigned int
max_cm_distance = 0);
static   unsigned   int   convert_cm(unsigned   int
echoTime);
static   unsigned   int   convert_in(unsigned   int
echoTime);
#if TIMER_ENABLED == true
void ping_timer(void (*userFunc)(void), unsigned int
max_cm_distance = 0);
boolean check_timer();
unsigned long ping_result;
static  void  timer_us(unsigned  int  frequency,  void
(*userFunc)(void));
static void timer_ms(unsigned long frequency, void
(*userFunc)(void));
static void timer_stop();
#endif
private:
boolean ping_trigger();
void set_max_distance(unsigned int max_cm_distance);
#if TIMER_ENABLED == true
boolean        ping_trigger_timer(unsigned        int
trigger_delay);
boolean ping_wait_timer();
static void timer_setup();
static void timer_ms_cntdwn();
#endif
#if  DO_BITWISE  ==  true  uint8_t  _triggerBit;uint8_t
_echoBit;
#if defined(PARTICLE)
#if !defined(portModeRegister)
#if defined (STM32F10X_MD)
#define portModeRegister(port)     ( &(port->CRL) )
#elif defined (STM32F2XX)
#define portModeRegister(port)     ( &(port->MODER)
#endif
#endif
volatile uint32_t *_triggerOutput;
volatile uint32_t *_echoInput;
volatile uint32_t *_triggerMode;
#else
volatile uint8_t *_triggerOutput;
volatile uint8_t *_echoInput;
volatile uint8_t *_triggerMode;
```

```
      #endif
      #else
      uint8_t _triggerPin;
      uint8_t _echoPin;
      #endif
      unsigned int _maxEchoTime;
      unsigned long _max_time;
      };
      #endif
```

Program pengujian 1 sensor

```
// ---------------------------------------------------------
      ----------------
// Example NewPing library sketch that does a ping about 20
      times per second.
// ---------------------------------------------------------
      ----------------#include <NewPing.h>
#define TRIGGER_PIN  3   // pin trigger terkoneksi ke arduino
#define ECHO_PIN     4  // pinecho terkoneksi pada arduino
#define MAX_DISTANCE 200 // jarak maksimal yang dibatasi oleh
      software untuk pembacaan sensor
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // set up
      pin yang digunakan di diproses pada library

void setup() {
  Serial.begin(115200);  //  kecepatan  serial(kecepatan  data
      untuk melakukan laporan pada komputer).}

void loop() {
  delay(100);// waktu tunggu sonar balik pada sensor(tdk boleh
      lebih kecil dari 29ms)satuan ms pembacaan ideal 100ms
  Serial.print("jarak : ");
  Serial.print(sonar.ping_cm());// proses pembacaan sensor
  Serial.println("cm") ;// proses waktu tunggu pelaporan pada
      komputer }
```

Program pengujian 3 sensor

```
// ---------------------------------------------------------
      ----------------
// Example NewPing library sketch that pings 3 sensors 20 times
      a second.
// ---------------------------------------------------------
      ----------------
#include <NewPing.h>
```

```
#define SONAR_NUM 3 // jumlah sensor (maksimal 4 sensor
        terpasang.
#define MAX_DISTANCE 200 // jarak maksimal pembacaan sensor
        yang dibatasi oleh program lebih dari jarak ini
        pembacaan dianggap 0cm
NewPing sonar[SONAR_NUM] = {// jumlah sensor dalam 1 paket
        pengiriman data.
NewPing(3, 4, MAX_DISTANCE),//sensor 1 // koneksi sensor
        terhadap arduino(trigger, echo)
NewPing(5, 6, MAX_DISTANCE), // sensor 2
NewPing(7, 8, MAX_DISTANCE)// sensor 3};


void setup() {
 Serial.begin(115200); // Open serial monitor at 115200 baud to
        see ping results.}

void loop() {
 delay(100);
 Serial.print("jarak sensor 1 =");
 Serial.print(sonar[0].ping_cm());//      memanggil      hasil
        perhitungan sensor 1
 Serial.print("cm \n");

 Serial.print("jarak sensor 2 =");
 Serial.print(sonar[1].ping_cm());//      memanggil      hasil
        perhitungan sensor 2
 Serial.print("cm \n");

  Serial.print("jarak sensor 3 =");
  Serial.print(sonar[2].ping_cm());//      memanggil      hasil
        perhitungan sensor 3
  Serial.print("cm \n");
  Serial.print("\n");
  delay(1000);}
```

Program pengujian servo motor

```
/* Sweep by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.
 modified 8 Nov 2013
 by Scott Fitzgerald
 https://www.arduino.cc/en/Tutorial/LibraryExamples/Sweep*/
#include <Servo.h>
Servo myservo;// memangging fungsi perhitungan servo
int pos = 0;// setting sudut awal servo
```

```
void setup() {
 myservo.attach(10);//  Pin  data  servo  yang  terkoneksi  ke
        arduino}

void loop() {
 myservo.write(0);//  mengatur  sudut  pergerakan  servo  yang
        diinginkan
delay(1000);  // waktu tunggu pergerakan ke sudut berikutnya
myservo.write(90);//  mengatur  sudut  pergerakan  servo  yang
        diinginkan
 delay(2000);// waktu tunggu pergerakan ke sudut berikutnya
myservo.write(180);//  mengatur  sudut  pergerakan  servo  yang
        diinginkan
delay(3000);// waktu tunggu pergerakan ke sudut berikutnya}
```