LAMPIRAN
Lampiran 1. Desain Robot
Lampiran 2. Desain Sistem

A: Catu Daya 1 (Enterai Li-Po 3s 1800 mAh)
B: Sistem Kontrol (Arduino Uno R3, Motor Driver L298N)
C: Output (Motor DC, Servo Sg90, Servo Mg90, Servo Mg996, LED, Buzzer)
D: Sistem Navigasi (Sensor Kompas)
E: Kontroler (Stik PS2, Receiver Stik PS2)
F: Catu Daya 2 (Baterai Li-Po 2s 1000 mAh, DC-DC Step Down)
G: Monitoring (Kamera FPV, Receiver ROTG, Smartphone, VR Box)
Lampiran 3. Foto Robot
Lampiran 4. Rangkaian Elektronik
Lampiran 5. Listing Program

```c
#include <PS2X.h>
#include <Arduino.h>
#include <IMC5883L_Simple.h>
#include <Wire.h>
#include <Servo.h>

#define MR1 3
#define MR2 5
#define ML1 6
#define ML2 9
#define buzzer A2
#define LED A3

PS2X ps2x;
Servo shaftServo;
Servo rotateYServo;
Servo rotateXServo;
Servo gripServo;
HM/C5883L_Simple Compass;

long periodMove = 50;
long periodNav = 50;
long periodED = 50;
long periodArm = 50;
long timeMove = 0;
long timeNav = 0;
long timeED = 0;
long timeArm = 0;
float sudutNav;

void indicatorSlapt(){
digitalWrite(buzzer, HIGH);
delay(50);
digitalWrite(buzzer, LOW);
delay(50);
digitalWrite(buzzer, HIGH);
delay(50);
digitalWrite(buzzer, LOW);
delay(50);
digitalWrite(buzzer, HIGH);
delay(50);
digitalWrite(buzzer, LOW);
}

void belokKiri(){
analogWrite(MR2, 238);
analogWrite(ML1, 255);
}

void robotStop(){
analogWrite(MR1, 0);
analogWrite(MR2, 0);
analogWrite(ML1, 0);
analogWrite(ML2, 0);
}

void robotNavigasi(){
analogWrite(MR2, 192);
analogWrite(ML1, 200);
}

void moveRobot(){
if(ps2x.Analog(PSS_LX)>64&&,(ps2x.Analog(PSS_LX)>=64)
&,&(ps2x.Analog(PSS_LX)<192))
{
   jauh();
}
else if(ps2x.Analog(PSS_LX)>192&&,(ps2x.Analog(PSS_LX)
>=64&&,(ps2x.Analog(PSS_LX)<192))
{
   belokKanun();
}
else if(ps2x.Analog(PSS_LX)>192&&,(ps2x.Analog(PSS_LX)
>=64&&,(ps2x.Analog(PSS_LX)<192))
{
   mundur();
}
else if(ps2x.Analog(PSS_LX)>64&&,(ps2x.Analog(PSS_LX)
>=64&&,(ps2x.Analog(PSS_LX)<192))
{
   belokKiri();
}
else{
   robotStop();
}

void sistemNavigasi(){
if(ps2x.Button(PSB_PAD_UP))
{
   if(sudutNav < 345 && sudutNav > 335)
   {
      robotStop();
   }
   else{
      robotNavigasi();
   }
}

void mundur(){
if(ps2x.Button(PSB_PAD_RIGHT))
{
   if(sudutNav > 82 && sudutNav < 92)
   {
      robotStop();
   }
   else{
      robotNavigasi();
   }
}

void belokKanun(){
analogWrite(MR2, 250);
analogWrite(ML2, 255);
}
```

LISTING PROGRAM

KETERANGAN

TEKNIK ELEKTROTEKNIK
UNIVERSITAS Negeri YOGYAKARTA

SKA. 1/1
DIP. FACHRUH. A

A4
NO. 5

DIG. R. A. RAJIF
DIB. 24/5/2019

NIM. 185710428

108
else if(ps2x.Button(PSB_PAD_DOWN)){
    if(tuduhNav > 203 && sudutNav < 215){
        robotStop();
    } else {
        robotNavigasi();
    }
}
else if(ps2x.Button(PSB_PAD_LEFT)){
    if(tuduhNav > 270 && sudutNav < 280){
        robotStop();
    } else {
        robotNavigasi();
    }
}
void lampu(){
    if(ps2x.Button(PSB_R)){
        digitalWrite(LED, HIGH);
    } else if(ps2x.Button(PSB_L)){
        digitalWrite(LED, LOW);
    }
    void servoSpeed(Servo servo, int sudut, uint8_t Speed){
        if(Speed == 0){
            return;
        }
        int posisiSekarang = servo.read();
        for(int i = posisiSekarang; i > sudut; i--){
            servo.write(i);
            if(Speed > 174){
                delayMicroseconds(256 - Speed) * 200;
            } else {
                delay((uint16_t)(256 - Speed) * 0.2);
            }
        }
        servo.write(sudut);
    }
    void arm(){
        if((ps2x.Analog(PSR_R) <= 0)){
            servoSpeed(shafServo, 163, 235);
        } else if((ps2x.Analog(PSR_R) <= 30)){
            servoSpeed(shafServo, 150, 235);
        } else if((ps2x.Analog(PSR_R) <= 60)){
            servoSpeed(shafServo, 130, 235);
        } else if((ps2x.Analog(PSS_R) <= 90)){
            servoSpeed(shafServo, 310, 235);
        }
        else if((ps2x.Analog(PSR_Y) <= 127)){
            servoSpeed(shiftServo, 95, 235);
        }
    }
    else if(ps2x.NewButtonState(PSB_BLUE)){
        rotateXServo.write(00);
    } else if(ps2x.ButtonPressed(PSB_RED)){
        rotateXServo.write(180);
    } else if(ps2x.ButtonReleased(PSB_PINK)){
        rotateXServo.write(0);
    }
    if(ps2x.Button(PSB_GREEN)){
        rotateYServo.write(10);
    } else {
        rotateYServo.write(88);
    }
    if(ps2x.Button(PSB_R2)){
        gripServo.write(45);
    } else if(ps2x.Button(PSB_L2)){
        gripServo.write(90);
    }
    void setup() {
        Serial.begin(115200);
        Wire.begin();
        ps2x.config_gamepad(10, 12, 11, 13, false, false);
        pinMode(MR1, OUTPUT);
        pinMode(MR2, OUTPUT);
        pinMode(ML1, OUTPUT);
        pinMode(ML2, OUTPUT);
        pinMode(buzzer, OUTPUT);
        pinMode(LED, OUTPUT);
        Compass.SetDeclination(0, 49, 'E');
        Compass.SetScaleMode(COMPASS_SCALE_130);
        Compass.SetOrientation(COMPASS_HORIZONTAL_X_NORTH);
        shaftServo.attach(2);
        shafServo.write(95);
        rotateYServo.attach(4);
        rotateYServo.write(88);
        rotateXServo.attach(7);
        rotateXServo.write(90);
        gripServo.attach(8);
        gripServo.write(90);
        delay(3000);
        robotStop();
        indikatorStop();
    }
}

<table>
<thead>
<tr>
<th>LISTING PROGRAM</th>
<th>KETERANGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A4</td>
</tr>
<tr>
<td></td>
<td>NO. 5</td>
</tr>
</tbody>
</table>

109
void loop() {
    ps2x.read_gamepad();
    rudderNav = Compass.GetHeadingDegrees();
    if(rudderNav > time_Move > periodMove){
        moveRobot();
        time_Move = millis();
    }
    if(rudderNav > periodNav){
        time_Nav = millis();
        sistemNavigasi();
    }
    if(time_LED > periodLED){
        time_LED = millis();
        lampu();
    }
    if(torque > time_Arm > periodArm){
        time_Arm = millis();
        arm();
    }
    delay(15);
}
Lampiran 6. Daftar Komponen

<table>
<thead>
<tr>
<th>No.</th>
<th>Komponen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Baterai Li-Po 3 Cell 1800 mAh</td>
</tr>
<tr>
<td>2</td>
<td>Baterai Li-Po 2 Cell 1000 mAh</td>
</tr>
<tr>
<td>3</td>
<td>Modul Regulator XL4005</td>
</tr>
<tr>
<td>4</td>
<td>Stik PS2 <em>Wireless</em></td>
</tr>
<tr>
<td>5</td>
<td>Sensor Kompas</td>
</tr>
<tr>
<td>6</td>
<td>Kamera FPV</td>
</tr>
<tr>
<td>7</td>
<td>Arduino Uno R3</td>
</tr>
<tr>
<td>8</td>
<td>Motor Driver L298N</td>
</tr>
<tr>
<td>9</td>
<td>Motor DC</td>
</tr>
<tr>
<td>10</td>
<td>Motor Servo</td>
</tr>
<tr>
<td>11</td>
<td>LED</td>
</tr>
<tr>
<td>12</td>
<td><em>Buzzer</em></td>
</tr>
<tr>
<td>13</td>
<td>Receiver Kamera FPV ROTG</td>
</tr>
<tr>
<td>14</td>
<td>VR Box</td>
</tr>
<tr>
<td>15</td>
<td>Box Panel</td>
</tr>
</tbody>
</table>

*DAFTAR KOMPONEN*                                      |  *KETERANGAN*  |
---                                                    |   |
TEKNIK ELektroNIS - DE                                      | A4   |
UNIVERSITAS SOEKARNO HATA                                      | NO. 6   |
SKA. 1:1                                                   |       |
DIP. FATHULA.                                              |       |
DIG. R. A. RAHF.                                            |       |
DIG. 24/6/2019                                             |       |
NIM. 18507124028                                           |       |

111
Lampiran 7. *Datasheet Arduino Uno R3*

**INTRODUCTION**

Arduino is used for building different types of electronic circuits easily using both a physical programmable circuit board usually microcontroller and piece of code running on computer with USB connection between the computer and Arduino.

Programming language used in Arduino is just a simplified version of C++ that can easily replace thousands of wires with words.
ARDUINO UNO-R3 PHYSICAL COMPONENTS

ATMEGA328P-PU microcontroller

The most important element in Arduino Uno R3 is ATMEGA328P-PU. It is an 8-bit Microcontroller with flash memory up to 32k bytes. It’s features as follow:

- High Performance, Low Power AVR
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - 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 program memory
  - 256/512/1024/2K Bytes EEPROM
  - 512/1K/2K/4K Bytes Internal RAM
  - Write/Erase Cycles: 10,000 Write/100,000 EEPROM
  - Data retention: 20 years at 85 °C / 100 years at 25°C
  - 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 SOIC Package
  - Temperature Measurement
  - Programmable Serial USART
Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (I2C 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
- 64 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-pin QFN/MLF and 32-pin QFN/MLF

- Operating Voltage:
  - 1.8 - 5.5V

- Temperature Range:
  - -40°C to 85°C

- Speed Grade:
  - 0 - 1 MHz at 1.8 - 5.5V, 0 - 10 MHz at 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V

- Power Consumption at 1 MHz, 1.8V, 25°C:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 μA
  - Power-save Mode: 0.75 μA (Including 32 kHz RTC)
Lampiran 8. Datasheet Motor Driver L298N
Connection Examples:

Controlling 2 DC Motor with +5V Arduino onboard Power Supply:

Below is the circuit connection use the on-board +5V power supply from Arduino board, and should be done without the 5V Enable Jumper on (Active 5V). This connection can drive two 5V DC motors simultaneously.

![Diagram of motor control setup](image)

Sketch Listing:

Copy and paste the sketch below to Arduino IDE and upload to Arduino Uno/Mega board.

```c
// Author: Hands on Technology
// Project: Arduino Uno
// Description: L298N Motor Driver
// Source Code: L298N_Motor.ino
// Program: Control 2 DC motors using L298N H Bridge Driver

// Definitions Arduino pins connected to input H Bridge
int IN1 = 8;
int IN2 = 9;
int IN3 = 10;
int IN4 = 11;

void setup()
{
  // Set the output pins

  // Code for motor control
}
```

www.handsontec.com
3-axis Digital Compass Module

Using a magnetometer can be a little tricky, especially if you’re unsure about the formulas to use to get the correct bearing and when other magnetic objects are interfering with your signal. We’ve created a library for our HMC5883L module, which will also be compatible with other HMC5883L breakout boards made by other manufacturers.

Join us whilst we cover the following:
• Understand what is a magnetometer and how they work.
• Introduce the HMC5883L Arduino Library
• Explain how to read data from the HMC5883L
• Explain how to calculate a bearing from this data.

So, assuming you have an HMC5883L Breakout Board and an Arduino, we will walk through using the HMC5883L breakout board to output our bearing.

Note: The HMC5883L is not the same as the HMC5863L. This library only covers the HMC5883L sold on our website.

How do compasses work?

Firstly, an introduction. A (standard handheld) compass works by aligning itself to the earth’s magnetic field. Because the compass’ needle is a ferrous material, it aligns itself to the magnetic field at the earth's pull. These magnetic fields are everywhere, and we use them to help us tell all directions we're facing.

Our magnetometer uses these magnetic fields, however, because it doesn’t, it can’t move a little needle inside it (it probably wouldn’t fit anyway). Inside our magnetometer are three magnetoresistive sensors on three axes. These can be quite complicated to understand (and to mention explained). It’s sufficient to say that the effect of magnetic fields on these sensors adjusts the current flow through the sensor. By applying a scale to this current, we can tell the magnetic force (measured in Gauss) on this sensor.
For a detailed explanation of the magnetor-rotational sensors, refer to the application note: "Magnetor-rotational Sensors".

By combining information about two or more of these axes, we can start to use the differences in the magnetic fields in three axes to infer the heading to magnetic north.

**How do we use one?**

Okay, so now we have to use one, the first step is to get some data from our compass. The HMC5883L is a device which communicates a wire that is a very easy communication protocol to use, and our favorite way to interface with our previous HMCs here at Love Electronic. All you need to do is plug the breakout board into your breadboard and connect up the following pins to your Arduino:

For Arduino Uno/Berlinino):
- Arduino GND -> HMC5883L GND
- Arduino VIN = HMC5883L VCC
- Arduino A0 (SG) -> HMC5883L SDI
- Arduino A1 (SCL) -> HMC5883L SDO

For Arduino Mega):
- Arduino GND -> HMC5883L GND
- Arduino VIN = HMC5883L VCC
- Arduino A0 (SG) -> HMC5883L SDI
- Arduino A1 (SCL) -> HMC5883L SDO

Note: You will also need two pull-up resistors to enable I2C. For this, connect two 22k- or 10k resistors between SDI and VCC, and SCL and VCC. We also tested it without pull-up resistors. It works well. For more information about I2C pull-up resistors, please visit this page: [http://arduino.cc/en/newbieguide/I2C](http://arduino.cc/en/newbieguide/I2C)
Now, of course to talk to the HMC5883L, we will need some code. Luckily we've written an Arduino library for the HMC5883L, which makes this really easy. Simply download the Arduino Library for HMC5883L and extract it to your Arduino folder in your Arduino installation. Mine is here: C:\Program Files (x86)\Arduino\libraries\HMC5883L\libraries

Once the library is installed start your Arduino IDE and we can get coding. If you just want to go and get all the code up without coding along, simply open the HMC5883L_Example file from the Arduino menu, otherwise we can code along together and you can fully understand everything we are going to write.

### Using the HMC5883L Arduino Library

What we want to accomplish is an Arduino sketch that will tell us the direction we are pointing in degrees, so it should read 0° for when we are pointing at magnetic north, and 180° when we are pointed south.

For a quick reality check, the bearing you get from your compass will be a little off, the compass senses magnetic fields, so any ferrous material anywhere near your compass can affect your output considerably. Also any kind of radio waves (I'm looking at you mobile phone) or don't even let it see any stereo speakers!

So, open your Arduino IDE and begin a new sketch, and quickly start off by writing the initial setup and loop method placeholders.
Then we must import the HMC5883L Arduino library for us to use. Because you installed the library into your Libraries folder, you should be able to add it by selecting the following menu option.

The HMC5883L communicates over I²C, so we must also import the Wire library (do this the same way) and add the following code to the setup() method to start the Wire library when we begin our Sketch:

```cpp
Wire.begin();
```

We will now be able to use devices on the I²C bus. We also want to communicate with our computer to report our findings, so initialize the serial port by adding the following two lines into setup():

```cpp
Serial.begin(9600);
Serial.print("Serial started.");
```

So, if you run the sketch we should see "Serial started." reported in the serial window. Great! Now we need to declare an instance of the HMC5883L, we can use throughout our sketch. So let’s add an HMC5883L as a global variable by adding outside the setup() and loop() methods, and initialize it inside the setup() method.

```cpp
// Add this outside the methods as a global variable.
HMC5883L compass;

// Add this at the end of setup() to create an instance of HMC5883L
compass = HMC5883L();
```

Once we have an instance of the compass we need to set it up, the compass needs to know what kind of gain (response scale) to work with, and how to output its measurements. We can configure the compass by adding the following lines after creating the compass in setup():

```cpp
Serial.println("Setting scale to +/- 1.3 Gauss");
if (error = compass.setScale(1.33)) // If there is an error, print it out.
    Serial.println(compass.getErrorText(error));
compass = MeasurementMode(Measurement_Coincide); // Set the measurement mode to Coincide.
compass.setMeasurementMode(0); // Set the measurement mode to Continuous;
```
125

This will set the gain of the device to 1.3 Gauss (Gs), this means the device is able to report magnetic fields up to ±1.3 Gauss. The lower you can make this number the more precise the compass will be, however if you have other objects interfering with the compass you may need to raise it to avoid overloading the sensor. Bear in mind that the compass can only accept certain gains, so check the datasheet to see what is available. If you choose an invalid gain the application will let you know by means of the error variable the Set method returns.

Your HMC5883L should now be configured and taking measurements, all that remains is to ask the sensor the data! To make sure sensors are on the same page, here is my current setup:

So the HMC5883L Arduino library can provide two different values for you to use. You can call either of the following:

```
// Retrieve the raw values from the compass (not scaled).
MagnetometerRaw raw = compass.ReadRaw();

// Retrieve the scaled values from the compass (scaled to the configured scale).
MagnetometerScaled scaled = compass.ReadScale();
```
Neitoi returns the values received straight from the magnetometer; if you are not interested in the actual magnetic strength of the field you can use this to return a MagnetometerRaw structure, which has three fields for you to access the values on each axis:

```c
MagnetometerRaw row = compass.read Values();
int xVal = row.XVal;
int yVal = row.YVal;
int zVal = row.ZVal;
```

The `read Values` function, as before, uses the same structure (called `MagnetometerScaled`), however these values are scaled to a range of `±1.35 Gauss` when we called `compass.setScales(1, 1, 1);`. You can use this data when you want to know the actual magnetic value each sensor is seeing.

### Calculate your bearing

Now we know how to use the `arduino` library to communicate over I2C to the HMC5883L triple axis magnetometer chip we can put it all together to calculate the bearing.

What we are going to do is add the following code to the `loop()` method that is going to retrieve the values from the device and do the actual bearing calculation:

```c
void loop()
{
    // Retrieve the raw values from the compass (not scaled).
    MagnetometerRaw row = compass.read Values();
    // Retrieve the scaled values from the compass (scaled to the configured scale).
    MagnetometerScaled scaled = compass.read Values();
    // Calculate heading when the magnetometer is level, then correct for signs of axes.
    float heading = atan2(scaled.XVal, scaled.YVal);
    // Correct for when signs are reversed.
    if (heading < 0) heading += 2 * PI;
    // Convert radians to degrees for readability.
    float headingDegrees = heading * 180 / PI;
    // Output the data via the serial port.
    Serial.print("xVal: "); Serial.println(xVal);
    Serial.print("yVal: "); Serial.println(yVal);
    Serial.print("zVal: "); Serial.println(zVal);
    Serial.print("heading: "); Serial.println(headingDegrees);
}
```

We also need to add the `output()` method we refer to at the end of this method:

```c
void output(MagnetometerRaw row, MagnetometerScaled scaled, float heading, float headingDegrees)
{
    // Output the data to the serial port.
    Serial.print("xVal: "); Serial.println(xVal);
    Serial.print("yVal: "); Serial.println(yVal);
    Serial.print("zVal: "); Serial.println(zVal);
    Serial.print("heading: "); Serial.println(headingDegrees);
}
```
You can now go ahead and run the sketch. Assuming you have got all the code in and connected your compass properly, you should now be getting the following being reported from your serial port:

```
You should be able to rotate your device and see the bearing rotate from 0 to 360 degrees! Try to keep the compass away from anything magnetic to avoid interference, but try holding something like your mobile phone next to the compass to see how bad the effect can be on the results.
```