DAFTAR PUSTAKA

Alamulhuda, M., Wahyudi, W., & Afrisal, H. (2023). PERANCANGAN GRASPING CONTROL PADA ROBOT TANGAN MENGGUNAKAN REINFORCEMENT LEARNING. Transient: Jurnal Ilmiah Teknik Elektro.

Ajang, N., & Hendriyawan, M. S. (2019). Implementasi ROS (Robot Operating System) Pada Sistem Kendali Jarak Jauh Robot Bergerak Jenis Non-holonomic [Manuskrip tak dipublikasikan]. Program Studi Teknik Elekto, Fakultas Informasi Teknologi dan Elektro, Universitas Teknologi Yogyakarta.

Anshar, Muh. (2010). Directed Learning-based Evolutionary Approach for Legged Robot Motion. Lambert Academic Publishing. Australia.

CuriousInventor. (2013, 16 Februari). How the Kinect Depth Sensor Works in 2 Minutes. https://www.youtube.com/watch?v=uq9SEJxZiUg

Dudek, M. J. (2008). Inertial Sensors, GPS, and Odometry. Dalam B. Siciliano & O. Khatib (Eds.), Springer Handbook of Robotics (hal. 477-490). Springer Berlin Heidelberg.

Firmansyah, R.A. (2015). Sistem Auto Docking Pada Service Robot Menggunakan Persepsi Visual.

Francis, K., & Bird, B. (2011, 28 Juni). Kinect Hackers Are Changing the Future of Robotics. Wired.com. https://www.wired.com/2011/06/mf_kinect/

Goris, Kristof. (2005). Autonomous Mobile Robot Mechanical Design. Ixelles: Vrije Universiteit Brussel.

Indrawan, Gede. (2008). Perancangan dan Implementasi Kecerdasan-Buatan Robot Pencari Jalur Berbasis Mikrokontroler Basic Stamp. Tesis, Universitas Indonesia.

Kinect. (2017, 7 Januari). Wikipedia. https://en.wikipedia.org/wiki/Kinect

Labbé, M., & Michaud, F. (2018). Long-term online multi-session graph-based SLAM with memory management. Autonomous Robots, 42, 1133-1150. Springer.

Labbé, M., & Michaud, F. (2019). RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics, 36, 416-446. Wiley Online Library.

Mahandi, Y.D. (2021). Deteksi Objek Untuk Robot Bergerak Menggunakan Kamera Omnidirectional Berbasis Fitur Warna.

MathWorks. (Tidak Tersedia). Localize TurtleBot Using Monte Carlo Localization. MathWorks. https://www.mathworks.com/help/nav/ug/localize-turtlebot-using-monte-carlo-localization.html

Pranoto, B., & Firdaus, A.R. (2021). Rancang Bangun Lengan Robot dengan Sistem Kontrol Otomatis dan Human Machine Interface untuk Mesin Operasional Industri Manufaktur. Jurnal Energi dan Teknologi Manufaktur (JETM).

Prasetyo, T.H., Siradjuddin, I., & Sungkono, S. (2021). Sistem Kendali Wall Following Pada Mobile Robot Dengan Penggerak Mekanum Menggunakan Metode Fuzzy. Jurnal Elektronika dan Otomasi Industri.

Riyanti, A., Pratiwi, N.D., Nainggolan, N.Y., Sardi, N.R., & Satrya, A.B. (2021). GLOBALISASI DAN TRANSFER TEKNOLOGI: PENOPANG INDUSTRI MANUFAKTUR PADA PERKEMBANGAN MARKETPLACE DI REGIONAL ASEAN.

Shobirin, M. (2016). Sistem Kendali Nirkabel Robot Bulutangkis Berbasis Mikrokontroller.

Sidiq, I.M., Ihsanto, E., & Yuliza, Y. (2020). Rancang Bangun Sistem Navigasi Robot Hexapod 3 DOF di Ruangan Labirin Menggunakan Metode Fuzzy Sugeno Berbasis Arduino.

Springer Handbook of Robotics. (2008). Edited by B. Siciliano and O. Khatib. Springer.

Supriyanto, R., Hustinawati., Nugraini, R. W., Kurniawan, A. B., Permadi, Y., Sa'ad, A. 2010. Robotika. Depok: Universitas Gunadarma.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic Robotics. The MIT Press.

Utomo, E.Y. (2015). Autonomous mobile robot berbasis landmark menggunakan particle filter dan occupancy grid maps untuk navigasi, penentuan posisi dan pemetaan.

Wicaksono, I. (2019). Navigasi Mobile Robot Darat Menggunakan Odometri Visual Berbasis Citra Stereo (Tesis Tugas Akhir, EE 184801). Institut Teknologi Sepuluh Nopember.

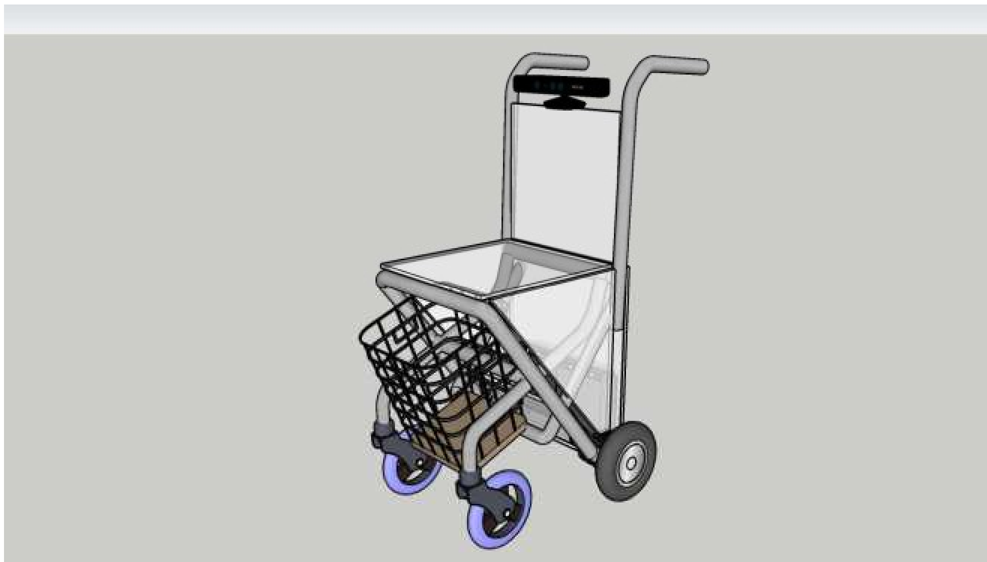
Zainuddin, N., Mustafah, Y., Shawgi, Y., & Rashid, N. M. (2014). Autonomous Navigation of Mobile Robot Using Kinect Sensor. Dalam International Conference on Computer & Communication Engineering, Kuala Lumpur, Malaysia.

Zamisyak, O., Prayogo, S.I., & Ahmad, B.S. (2016). Educational Multifunction Robot (Indobot) sebagai Robot Edukasi.

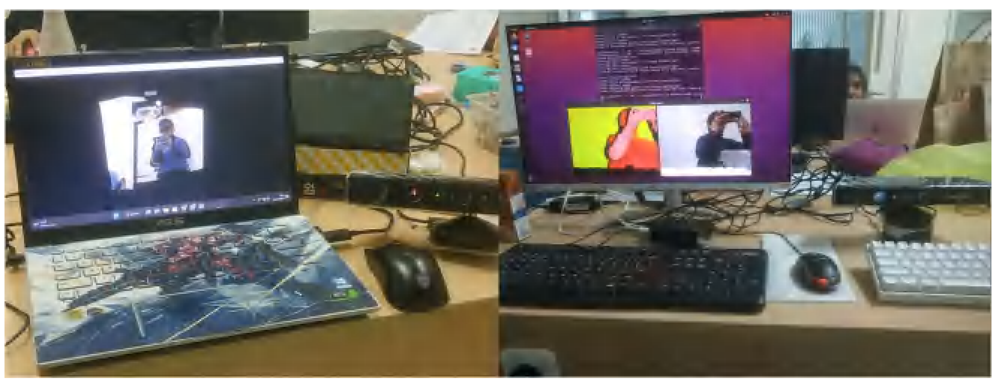
LAMPIRAN

Lampiran 1 Dokumentasi Pelaksanaan Penelitaian

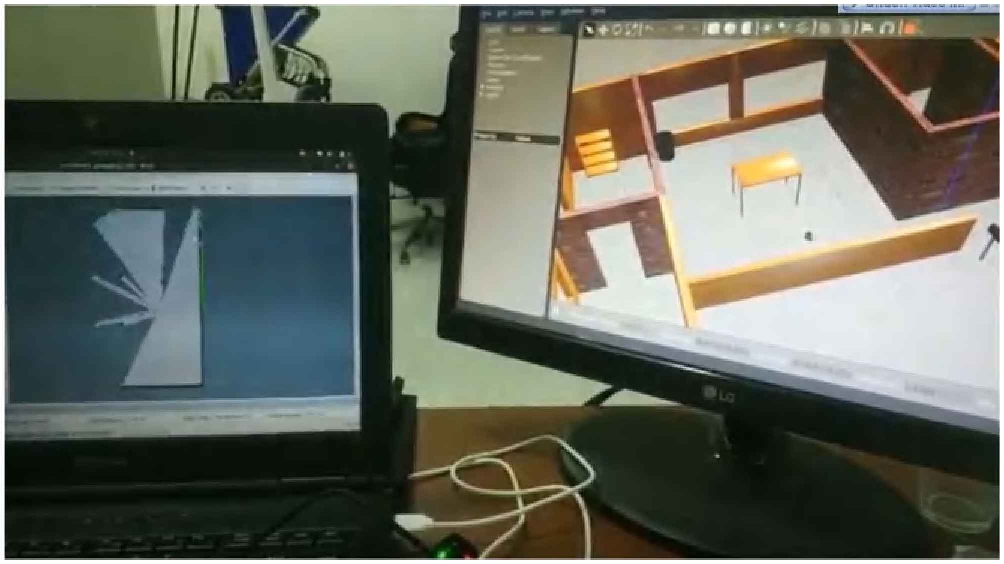
a. Tahap Perancangan



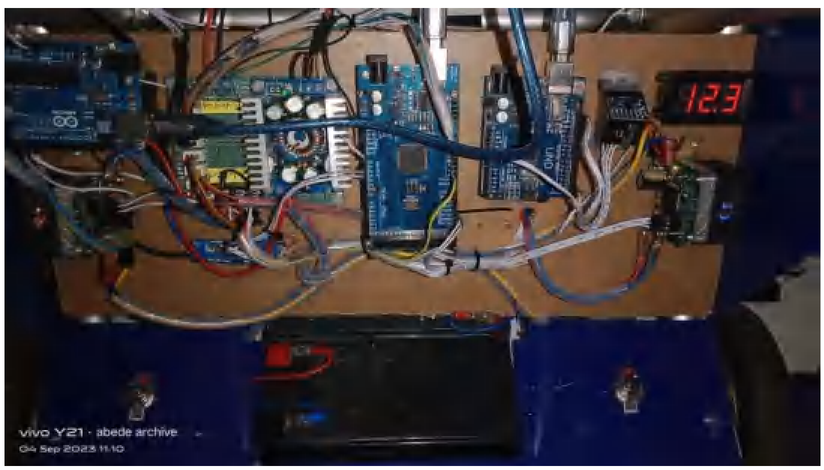
Rancangan Awal Desain Pada Software Sketchup dan Pemodelan URDF



Modifikasi Kamera Xbox Kinect Untuk Aplikasi Pada Robot



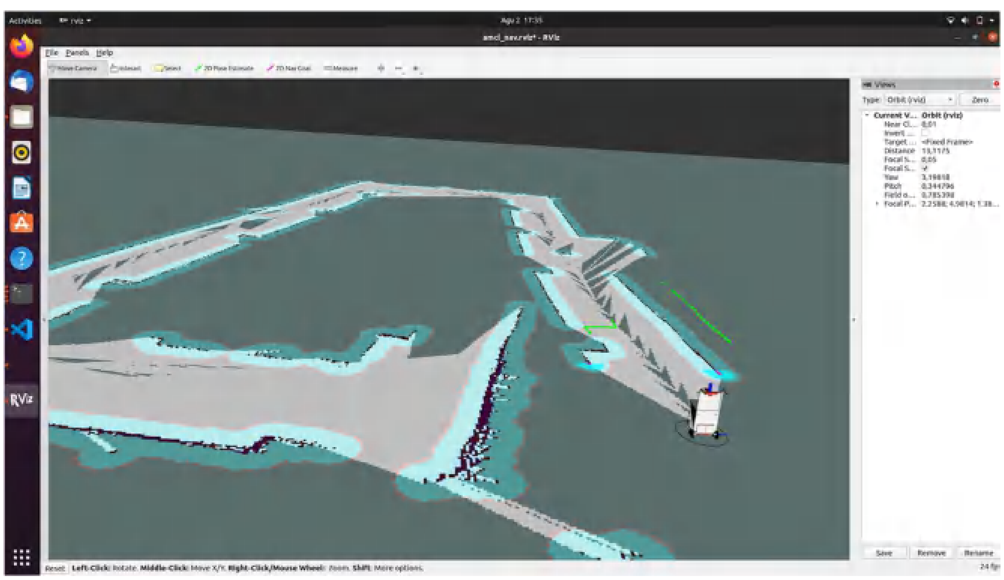
Pemodelan Algoritma SLAM dengan Software Simulasi Gazebo



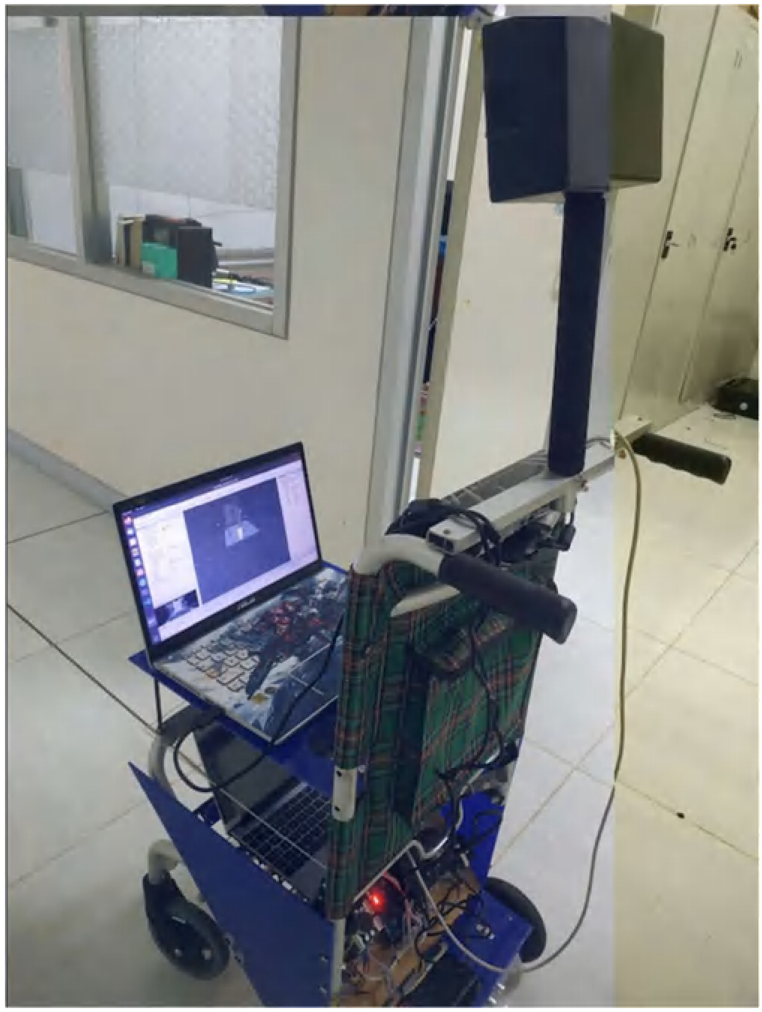
Instalasi perangkat keras

b. Tahap Pengujian

Pengujian Estimasi Odometry Robot Mengikuti Jalur yang Telah ditetapkan



Pengujian Pemetaan Koridor Lantai 2 Gedung COT



Pengujian Pemetaan Lab Sistem Kendali dan Instrumentasi

c. Video Pengujian

https://www.youtube.com/watch?si=DEtySbQt1YJKWk0g&v=tmyqf-2GT2k&feature=youtu.be

Lampiran 2 Kode Pemrograman Low Level

a. Kendali Motor dan Pembacaan Encoder

```
#include <ros.h>
#include <std_msgs/Int16.h>

// Handles startup and shutdown of ROS
ros::NodeHandle nh;

// Encoder output to Arduino Interrupt pin. Tracks the tick count.
#define ENC_IN_LEFT_A 2
#define ENC_IN_RIGHT_A 19

// Other encoder output to Arduino to keep track of wheel direction
// Tracks the direction of rotation.
#define ENC_IN_LEFT_B 3
#define ENC_IN_RIGHT_B 18

// True = Forward; False = Reverse
boolean Direction_left = true;
boolean Direction_right = true;

// Minumum and maximum values for 16-bit integers
const int encoder_minimum = -32768;
const int encoder_maximum = 32767;

// Keep track of the number of wheel ticks
std_msgs::Int16 right_wheel_tick_count;
ros::Publisher rightPub("right_ticks", &right_wheel_tick_count);

std_msgs::Int16 left_wheel_tick_count;
```

```
ros::Publisher leftPub("left_ticks", &left_wheel_tick_count);

// 100ms interval for measurements
const int interval = 100;
long previousMillis = 0;
long currentMillis = 0;

// Increment the number of ticks
void right_wheel_tick() {

  // Read the value for the encoder for the right wheel
  int val = digitalRead(ENC_IN_RIGHT_B);

  if(val == LOW) {
    Direction_right = false; // Reverse
  }
  else {
    Direction_right = true; // Forward
  }

  if (Direction_right) {

    if (right_wheel_tick_count.data == encoder_maximum) {
      right_wheel_tick_count.data = encoder_minimum;
    }
    else {
      right_wheel_tick_count.data++;
    }
  }
  else {
```

```
    if (right_wheel_tick_count.data == encoder_minimum) {
      right_wheel_tick_count.data = encoder_maximum;
    }
    else {
      right_wheel_tick_count.data--;
    }
  }
}

// Increment the number of ticks
void left_wheel_tick() {

  // Read the value for the encoder for the left wheel
  int val = digitalRead(ENC_IN_LEFT_B);

  if(val == LOW) {
    Direction_left = true; // Reverse
  }
  else {
    Direction_left = false; // Forward
  }

  if (Direction_left) {
    if (left_wheel_tick_count.data == encoder_maximum) {
      left_wheel_tick_count.data = encoder_minimum;
    }
    else {
      left_wheel_tick_count.data++;
    }
  }
```

```
  else {
    if (left_wheel_tick_count.data == encoder_minimum) {
      left_wheel_tick_count.data = encoder_maximum;
    }
    else {
      left_wheel_tick_count.data--;
    }
  }
}

void setup() {

  // Set pin states of the encoder
  pinMode(ENC_IN_LEFT_A , INPUT_PULLUP);
  pinMode(ENC_IN_LEFT_B , INPUT);
  pinMode(ENC_IN_RIGHT_A , INPUT_PULLUP);
  pinMode(ENC_IN_RIGHT_B , INPUT);

  // Every time the pin goes high, this is a tick
  attachInterrupt(digitalPinToInterrupt(ENC_IN_LEFT_A), left_wheel_tick, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_IN_RIGHT_A), right_wheel_tick, RISING);

  // ROS Setup
  nh.getHardware()->setBaud(115200);
  nh.initNode();
  nh.advertise(rightPub);
  nh.advertise(leftPub);
}

void loop() {
```

```
  // Record the time
  currentMillis = millis();

  // If 100ms have passed, print the number of ticks
  if (currentMillis - previousMillis > interval) {

    previousMillis = currentMillis;

    rightPub.publish( &right_wheel_tick_count );
    leftPub.publish( &left_wheel_tick_count );
    nh.spinOnce();
  }
}
```

b. Interface MPU6050

```
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation board)
```

```
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

/*
===============================================================
==========
  NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
  depends on the MPU-6050's INT pin being connected to the Arduino's
  external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
  digital I/O pin 2.
 *
===============================================================
========== */

/*
===============================================================
==========
  NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
  when using Serial.write(buf, len). The Teapot output uses this method.
  The solution requires a modification to the Arduino USBAPI.h file, which
  is fortunately simple, but annoying. This will be fixed in the next IDE
  release. For more info, see these links:

  http://arduino.cc/forum/index.php/topic,109987.0.html
  http://code.google.com/p/arduino/issues/detail?id=958
 *
===============================================================
========== */



// uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual
```

```
// quaternion components in a [w, x, y, z] format (not best for parsing
// on a remote host such as Processing or something though)
#define OUTPUT_READABLE_QUATERNION

// uncomment "OUTPUT_READABLE_EULER" if you want to see Euler angles
// (in degrees) calculated from the quaternions coming from the FIFO.
// Note that Euler angles suffer from gimbal lock (for more info, see
// http://en.wikipedia.org/wiki/Gimbal_lock)
//#define OUTPUT_READABLE_EULER

// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for
// more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
//#define OUTPUT_READABLE_YAWPITCHROLL

// uncomment "OUTPUT_READABLE_REALACCEL" if you want to see acceleration
// components with gravity removed. This acceleration reference frame is
// not compensated for orientation, so +X is always +X according to the
// sensor, just without the effects of gravity. If you want acceleration
// compensated for orientation, us OUTPUT_READABLE_WORLDACCEL instead.
//#define OUTPUT_READABLE_REALACCEL

// uncomment "OUTPUT_READABLE_WORLDACCEL" if you want to see acceleration
// components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no magnetometer
// is present in this case). Could be quite handy in some cases.
//#define OUTPUT_READABLE_WORLDACCEL
```

```
// uncomment "OUTPUT_TEAPOT" if you want output that matches the
// format used for the InvenSense teapot demo
//#define OUTPUT_TEAPOT




#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
/*
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
```

```
*/

#include <ros.h>
#include <ros/time.h>
#include <geometry_msgs/Quaternion.h>
ros::NodeHandle nh;
geometry_msgs::Quaternion quat_msg;
ros::Publisher quat_pub("quaternion", &quat_msg);

//
===================================================================
=
// ===          INTERRUPT DETECTION ROUTINE          ===
//
===================================================================
=

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}

//
===================================================================
=
// ===               INITIAL SETUP               ===
//
===================================================================
=

void setup() {
```

```
  // join I2C bus (I2Cdev library doesn't do this automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif

  // initialize serial communication
  // (115200 chosen because it is required for Teapot Demo output, but it's
  // really up to you depending on your project)
  Serial.begin(115200);
  while (!Serial); // wait for Leonardo enumeration, others continue immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3V or Arduino
  // Pro Mini running at 3.3V, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.

  // initialize device
  //Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);
/*
  // verify connection
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
```

```
    // wait for ready
    Serial.println(F("\nSend any character to begin DMP programming and demo: "));
    while (Serial.available() && Serial.read()); // empty buffer
    while (!Serial.available());                 // wait for data
    while (Serial.available() && Serial.read()); // empty buffer again

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
  */
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(228);
    mpu.setYGyroOffset(93);
    mpu.setZGyroOffset(-64);
    mpu.setZAccelOffset(1421); // 1688 factory default for my test chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
      // Calibration Time: generate offsets and calibrate our MPU6050
      mpu.CalibrateAccel(6);
      mpu.CalibrateGyro(6);
      mpu.PrintActiveOffsets();
      // turn on the DMP, now that it's ready
      //Serial.println(F("Enabling DMP..."));
      mpu.setDMPEnabled(true);

      // enable Arduino interrupt detection
      //Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
```

```
        //Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
        digitalPinToInterrupt(INTERRUPT_PIN);
        //Serial.println(F(")..."));
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows it's okay to use it
        //Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);

    nh.initNode();
    nh.advertise(quat_pub);
}
```

```
// =================================================================
=
// ===                    MAIN PROGRAM LOOP                   ===
// =================================================================
=

void loop() {

    nh.spinOnce();

    // if programming failed, don't try to do anything
    if (!dmpReady) return;
    // read a packet from FIFO
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet
        #ifdef OUTPUT_READABLE_QUATERNION
            // display quaternion values in easy matrix form: w x y z
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            /*
            Serial.print("quat\t");
            Serial.print(q.w);
            Serial.print("\t");
            Serial.print(q.x);
            Serial.print("\t");
            Serial.print(q.y);
            Serial.print("\t");
            Serial.println(q.z);
            */
            quat_msg.w = q.w;
```

```
    quat_msg.x = q.x;
    quat_msg.y = q.y;
    quat_msg.z = q.z;

    quat_pub.publish(&quat_msg);

#endif

#ifdef OUTPUT_READABLE_EULER
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetEuler(euler, &q);
    Serial.print("euler\t");
    Serial.print(euler[0] * 180/M_PI);
    Serial.print("\t");
    Serial.print(euler[1] * 180/M_PI);
    Serial.print("\t");
    Serial.println(euler[2] * 180/M_PI);
#endif

#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    q.y = -q.y;
    q.z = -q.z;
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    Serial.print("ypr\t");
    Serial.print(ypr[0] * 180/M_PI);
    Serial.print("\t");
```

```
    Serial.print(ypr[1] * 180/M_PI);
    Serial.print("\t");
    Serial.println(ypr[2] * 180/M_PI);
#endif

#ifdef OUTPUT_READABLE_REALACCEL
    // display real acceleration, adjusted to remove gravity
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    Serial.print("areal\t");
    Serial.print(aaReal.x);
    Serial.print("\t");
    Serial.print(aaReal.y);
    Serial.print("\t");
    Serial.println(aaReal.z);
#endif

#ifdef OUTPUT_READABLE_WORLDACCEL
    // display initial world-frame acceleration, adjusted to remove gravity
    // and rotated based on known orientation from quaternion
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
    Serial.print("aworld\t");
    Serial.print(aaWorld.x);
    Serial.print("\t");
```

```
            Serial.print(aaWorld.y);
            Serial.print("\t");
            Serial.println(aaWorld.z);
        #endif

        #ifdef OUTPUT_TEAPOT
            // display quaternion values in InvenSense Teapot demo format:
            teapotPacket[2] = fifoBuffer[0];
            teapotPacket[3] = fifoBuffer[1];
            teapotPacket[4] = fifoBuffer[4];
            teapotPacket[5] = fifoBuffer[5];
            teapotPacket[6] = fifoBuffer[8];
            teapotPacket[7] = fifoBuffer[9];
            teapotPacket[8] = fifoBuffer[12];
            teapotPacket[9] = fifoBuffer[13];
            Serial.write(teapotPacket, 14);
            teapotPacket[11]++; // packetCount, loops at 0xFF on purpose
        #endif

        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
    }
}
```

Lampiran 3 Pemodelan Robot dengan URDF

```
<?xml version="1.0"?>

<robot name="mobile_robot" xmlns:xacro="http://ros.org/wiki/xacro">


  <material name="blue">

        <color rgba="0 0 0.8 1"/>

  </material>


  <material name="white">

        <color rgba="1 1 1 1"/>

  </material>


  <material name="black">

        <color rgba="0 0 0 1"/>

  </material>


  <material name="green">

    <color rgba="0.0 0.8 0.0 1.0"/>

  </material>


  <material name="orange">

    <color rgba="1 0.3 0.1 1"/>

  </material>


  <material name="red">

    <color rgba="1 0 0 1"/>

  </material>


  <link name="base_link">

    <!--<visual>
```

```
        <geometry>
          <box size="0.24 0.24 0.001"/>
        </geometry>
        <material name="white"/>
        <origin rpy="0 0 0" xyz="0 0 0.0005"/>
        <material>
          <color rgba="0 0 0.8 1" />
        </material>
      </visual> -->
      <collision>
        <origin rpy="0 0 0"/>
        <geometry>
          <box size="0.49 0.41 0.83"/>
        </geometry>
      </collision>
    </link>

    <joint name="base_link_joint" type="fixed">
      <origin xyz="-0.1 0 0"/>
      <parent link="base_link" />
      <child link="base_plate" />
    </joint>

    <link name="base_plate">
    <visual>
      <origin xyz="0.53 0.213 -0.1" rpy="0 0 ${pi}"/>
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/main_body.stl"
  scale="0.001 0.001 0.001" />
      </geometry>
```

```
    <material name="white"/>
  </visual>
  <collision>
    <origin xyz="0.53 0.213 -0.1" rpy="0 0 ${pi}"/>
    <geometry>
      <mesh filename="package://autonomus_mobile_robot/stl/main_body.stl" scale="0.001 0.001 0.001" />
    </geometry>
  </collision>
  </link>

  <!-- RIGHT WHEEL LINK -->
  <joint name="right_wheel_joint" type="continuous">
    <origin xyz="0 -0.23 0" rpy="${pi/2} 0 0" />
    <parent link="base_plate" />
    <child link="right_wheel" />
    <axis xyz="0 0 -1"/>
  </joint>

  <link name="right_wheel">
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <mesh filename="package://autonomus_mobile_robot/stl/wheel.stl" scale="0.001 0.001 0.001"/>
    </geometry>
    <material name="black"/>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
```

```
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/wheel.stl" scale="0.001 0.001 0.001" />
      </geometry>
    </collision>
    </link>

    <!-- LEFT WHEEL LINK -->
    <joint name="left_wheel_joint" type="continuous">
        <origin xyz="0 0.23 0" rpy="0 ${pi} ${pi/2}"/>
        <parent link="base_plate" />
        <child link="left_wheel" />
        <axis xyz="-1 0 0" />
    </joint>

    <link name="left_wheel">
    <visual>
      <origin xyz="0 0 0" rpy="${pi} -${pi/2} 0"/>
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/wheel.stl" scale="0.001 0.001 0.001"/>
      </geometry>
      <material name="black"/>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="${pi} -${pi/2} 0"/>
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/wheel.stl" scale="0.001 0.001 0.001" />
      </geometry>
    </collision>
```

```
    </link>


    <!-- KINECT LINK -->
    <joint name="laser_joint" type="fixed">
        <origin xyz="-0 0 0.83" rpy="0 0 0"/>
        <parent link="base_plate" />
        <child link="kinect_link" />
    </joint>


    <link name="kinect_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 ${pi}"/>
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/kinect.stl" scale="0.001
0.001 0.001"/>
      </geometry>
      <material name="orange"/>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 ${pi}"/>
      <geometry>
        <mesh filename="package://autonomus_mobile_robot/stl/kinect.stl" scale="0.001
0.001 0.001" />
      </geometry>
    </collision>
    </link>


</robot>
```

Lampiran 4 Kode Pemrograman ROS

a. node imu quaternion converter

```
#include <ros/ros.h>
#include <geometry_msgs/Quaternion.h>
#include <geometry_msgs/PoseStamped.h>
#include <sensor_msgs/Imu.h>

ros::Publisher imuPub_;
ros::Publisher posePub_;

geometry_msgs::PoseStamped pose_msg;

sensor_msgs::Imu imu_msg;

void quaternionCb(const geometry_msgs::Quaternion::ConstPtr& msg){

  bool pose_subscribed = (posePub_.getNumSubscribers() > 0);
  bool imu_subscribed  = (imuPub_.getNumSubscribers() > 0);

  if (pose_subscribed || imu_subscribed){

    //@TODO make time offset param
    ros::Time sensor_data_time = ros::Time::now() - ros::Duration(0.002);

    if (imu_subscribed){
      imu_msg.header.stamp = sensor_data_time;

      imu_msg.orientation = *msg;

      imuPub_.publish(imu_msg);
```

```
    }

    if (pose_subscribed){
      pose_msg.header.stamp = sensor_data_time;

      pose_msg.pose.orientation = *msg;

      posePub_.publish(pose_msg);
    }
  }
}

int main(int argc, char** argv){

    ros::init(argc, argv, "mpu6050_imu_converter");

    ros::NodeHandle _nh;
    ros::NodeHandle _pnh("~");

    double linear_acceleration_stdev, angular_velocity_stdev;
    std::string frame_id;
    std::string frame_id2;

    ros::param::param<double>("~linear_acceleration_stdev", linear_acceleration_stdev,
0.06);
    ros::param::param<double>("~angular_velocity_stdev", angular_velocity_stdev,
0.005);
    ros::param::param<std::string>("~frame_id", frame_id, std::string("imu_link"));

    double linear_acceleration_cov = linear_acceleration_stdev * linear_acceleration_stdev;
```

```
    double angular_velocity_cov = angular_velocity_stdev * angular_velocity_stdev;



    imu_msg.header.frame_id = frame_id;
    pose_msg.header.frame_id = frame_id;

    imu_msg.orientation_covariance[0] = 0.1;
    imu_msg.orientation_covariance[4] = 0.1;
    imu_msg.orientation_covariance[8] = 0.1;



    // Angular velocity entries are not filled, so set to -1.0
    imu_msg.angular_velocity_covariance[0] = 0.0; //angular_velocity_cov;
    imu_msg.angular_velocity_covariance[4] = angular_velocity_cov;
    imu_msg.angular_velocity_covariance[8] = angular_velocity_cov;

    // Linear acceleration entries are not filled, so set to -1.0
    imu_msg.linear_acceleration_covariance[0] = 0.0;
    imu_msg.linear_acceleration_covariance[4] = linear_acceleration_cov;
    imu_msg.linear_acceleration_covariance[8] = linear_acceleration_cov;

    posePub_ = _pnh.advertise<geometry_msgs::PoseStamped>("pose", 10);
    imuPub_ = _pnh.advertise<sensor_msgs::Imu>("imu", 10);

    ros::Subscriber quatSub_ = _nh.subscribe("quaternion", 10, quaternionCb);

    ros::spin();

    return(0);
}
```

b. node teleoperation joystick

```
#!/usr/bin/env python3


import rospy

from geometry_msgs.msg import Twist

from sensor_msgs.msg import Joy


def joy_callback(data):

    twist = Twist()

    twist.linear.x = data.axes[1] * 0.5 # Scale joystick input to half of maximum linear speed

    twist.angular.z = data.axes[0] * 0.5 # Scale joystick input to half of maximum angular speed

    pub.publish(twist)


rospy.init_node('joystick_controller')

pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)

rospy.Subscriber('joy', Joy, joy_callback)

rospy.spin()
```

c. node ektended Kalman filter odometry publisher

```
// Include various libraries

#include "ros/ros.h"

#include "std_msgs/Int16.h"

#include <nav_msgs/Odometry.h>

#include <geometry_msgs/PoseStamped.h>

#include <tf2/LinearMath/Quaternion.h>

#include <tf2_ros/transform_broadcaster.h>

#include <cmath>
```

```
// Create odometry data publishers
ros::Publisher odom_data_pub;
ros::Publisher odom_data_pub_quat;
nav_msgs::Odometry odomNew;
nav_msgs::Odometry odomOld;

// Initial pose
const double initialX = 0.0;
const double initialY = 0.0;
const double initialTheta = 0.00000000001;
const double PI = 3.141592;

// Robot physical constants
const double TICKS_PER_REVOLUTION = 36; // For reference purposes.
const double WHEEL_RADIUS = 0.15; // Wheel radius in meters
const double WHEEL_BASE = 0.46; // Center of left tire to center of right tire
const double TICKS_PER_METER = 76.433121; // Original was 2800

// Distance both wheels have traveled
double distanceLeft = 0;
double distanceRight = 0;

// Flag to see if initial pose has been received
bool initialPoseRecieved = false;

using namespace std;

// Get initial_2d message from either Rviz clicks or a manual pose publisher
void set_initial_2d(const geometry_msgs::PoseStamped &rvizClick) {
```

```
  odomOld.pose.pose.position.x = rvizClick.pose.position.x;
  odomOld.pose.pose.position.y = rvizClick.pose.position.y;
  odomOld.pose.pose.orientation.z = rvizClick.pose.orientation.z;
  initialPoseRecieved = true;
}

// Calculate the distance the left wheel has traveled since the last cycle
void Calc_Left(const std_msgs::Int16& leftCount) {

  static int lastCountL = 0;
  if(leftCount.data != 0 && lastCountL != 0) {

    int leftTicks = (leftCount.data - lastCountL);

    if (leftTicks > 10000) {
      leftTicks = 0 - (65535 - leftTicks);
    }
    else if (leftTicks < -10000) {
      leftTicks = 65535-leftTicks;
    }
    else{}
    distanceLeft = leftTicks/TICKS_PER_METER;
  }
  lastCountL = leftCount.data;
}

// Calculate the distance the right wheel has traveled since the last cycle
void Calc_Right(const std_msgs::Int16& rightCount) {

  static int lastCountR = 0;
```

```
  if(rightCount.data != 0 && lastCountR != 0) {

    int rightTicks = rightCount.data - lastCountR;

    if (rightTicks > 10000) {
      distanceRight = (0 - (65535 - distanceRight))/TICKS_PER_METER;
    }
    else if (rightTicks < -10000) {
      rightTicks = 65535 - rightTicks;
    }
    else{}
    distanceRight = rightTicks/TICKS_PER_METER;
  }
  lastCountR = rightCount.data;
}

// Publish a nav_msgs::Odometry message in quaternion format
void publish_quat() {

  tf2::Quaternion q;

  q.setRPY(0, 0, odomNew.pose.pose.orientation.z);

  nav_msgs::Odometry quatOdom;
  quatOdom.header.stamp = odomNew.header.stamp;
  quatOdom.header.frame_id = "odom";
  quatOdom.child_frame_id = "base_link";
  quatOdom.pose.pose.position.x = odomNew.pose.pose.position.x;
  quatOdom.pose.pose.position.y = odomNew.pose.pose.position.y;
  quatOdom.pose.pose.position.z = odomNew.pose.pose.position.z;
```

```
  quatOdom.pose.pose.orientation.x = q.x();
  quatOdom.pose.pose.orientation.y = q.y();
  quatOdom.pose.pose.orientation.z = q.z();
  quatOdom.pose.pose.orientation.w = q.w();
  quatOdom.twist.twist.linear.x = odomNew.twist.twist.linear.x;
  quatOdom.twist.twist.linear.y = odomNew.twist.twist.linear.y;
  quatOdom.twist.twist.linear.z = odomNew.twist.twist.linear.z;
  quatOdom.twist.twist.angular.x = odomNew.twist.twist.angular.x;
  quatOdom.twist.twist.angular.y = odomNew.twist.twist.angular.y;
  quatOdom.twist.twist.angular.z = odomNew.twist.twist.angular.z;

  for(int i = 0; i<36; i++) {
    if(i == 0 || i == 7 || i == 14) {
      quatOdom.pose.covariance[i] = .01;
     }
     else if (i == 21 || i == 28 || i== 35) {
       quatOdom.pose.covariance[i] += 0.1;
     }
     else {
       quatOdom.pose.covariance[i] = 0;
     }
  }

  odom_data_pub_quat.publish(quatOdom);
}

// Update odometry information
void update_odom() {

  // Calculate the average distance
```

```
double cycleDistance = (distanceRight + distanceLeft) / 2;

// Calculate the number of radians the robot has turned since the last cycle
double cycleAngle = asin((distanceRight-distanceLeft)/WHEEL_BASE);

// Average angle during the last cycle
double avgAngle = cycleAngle/2 + odomOld.pose.pose.orientation.z;

if (avgAngle > PI) {
  avgAngle -= 2*PI;
}
else if (avgAngle < -PI) {
  avgAngle += 2*PI;
}
else{}

// Calculate the new pose (x, y, and theta)
odomNew.pose.pose.position.x = odomOld.pose.pose.position.x + cos(avgAngle)*cycleDistance;
odomNew.pose.pose.position.y = odomOld.pose.pose.position.y + sin(avgAngle)*cycleDistance;
odomNew.pose.pose.orientation.z = cycleAngle + odomOld.pose.pose.orientation.z;

// Prevent lockup from a single bad cycle
if (isnan(odomNew.pose.pose.position.x) || isnan(odomNew.pose.pose.position.y)
   || isnan(odomNew.pose.pose.position.z)) {
  odomNew.pose.pose.position.x = odomOld.pose.pose.position.x;
  odomNew.pose.pose.position.y = odomOld.pose.pose.position.y;
  odomNew.pose.pose.orientation.z = odomOld.pose.pose.orientation.z;
}
```

```
  // Make sure theta stays in the correct range
  if (odomNew.pose.pose.orientation.z > PI) {
    odomNew.pose.pose.orientation.z -= 2 * PI;
  }
  else if (odomNew.pose.pose.orientation.z < -PI) {
    odomNew.pose.pose.orientation.z += 2 * PI;
  }
  else{}

  // Compute the velocity
  odomNew.header.stamp = ros::Time::now();
  odomNew.twist.twist.linear.x = cycleDistance/(odomNew.header.stamp.toSec() - odomOld.header.stamp.toSec());
  odomNew.twist.twist.angular.z = cycleAngle/(odomNew.header.stamp.toSec() - odomOld.header.stamp.toSec());

  // Save the pose data for the next cycle
  odomOld.pose.pose.position.x = odomNew.pose.pose.position.x;
  odomOld.pose.pose.position.y = odomNew.pose.pose.position.y;
  odomOld.pose.pose.orientation.z = odomNew.pose.pose.orientation.z;
  odomOld.header.stamp = odomNew.header.stamp;

  // Publish the odometry message
  odom_data_pub.publish(odomNew);
}

int main(int argc, char **argv) {

  // Set the data fields of the odometry message
```

```
odomNew.header.frame_id = "odom";
odomNew.pose.pose.position.z = 0;
odomNew.pose.pose.orientation.x = 0;
odomNew.pose.pose.orientation.y = 0;
odomNew.twist.twist.linear.x = 0;
odomNew.twist.twist.linear.y = 0;
odomNew.twist.twist.linear.z = 0;
odomNew.twist.twist.angular.x = 0;
odomNew.twist.twist.angular.y = 0;
odomNew.twist.twist.angular.z = 0;
odomOld.pose.pose.position.x = initialX;
odomOld.pose.pose.position.y = initialY;
odomOld.pose.pose.orientation.z = initialTheta;

// Launch ROS and create a node
ros::init(argc, argv, "ekf_odom_pub");
ros::NodeHandle node;

// Subscribe to ROS topics
ros::Subscriber subForRightCounts = node.subscribe("right_ticks", 100, Calc_Right, ros::TransportHints().tcpNoDelay());
ros::Subscriber subForLeftCounts = node.subscribe("left_ticks", 100, Calc_Left, ros::TransportHints().tcpNoDelay());
ros::Subscriber subInitialPose = node.subscribe("initial_2d", 1, set_initial_2d);

// Publisher of simple odom message where orientation.z is an euler angle
odom_data_pub = node.advertise<nav_msgs::Odometry>("odom_data_euler", 100);

// Publisher of full odom message where orientation is quaternion
odom_data_pub_quat = node.advertise<nav_msgs::Odometry>("odom_data_quat", 100);
```

```
  ros::Rate loop_rate(30);

  while(ros::ok()) {

    if(initialPoseRecieved) {
      update_odom();
      publish_quat();
    }
    ros::spinOnce();
    loop_rate.sleep();
  }

  return 0;
}
```

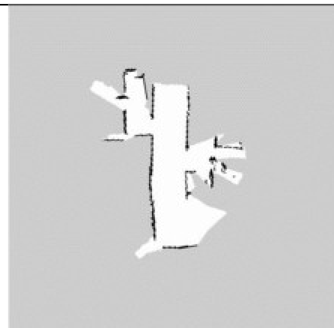
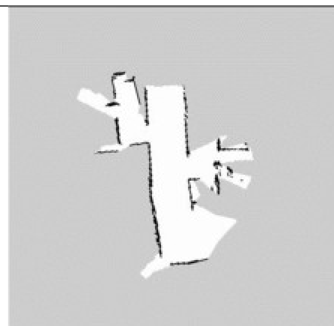
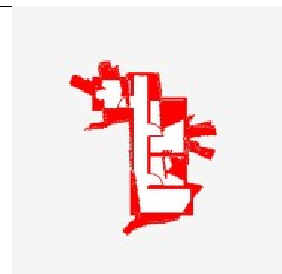
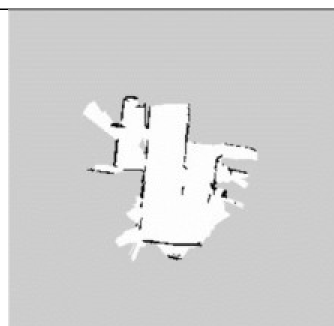
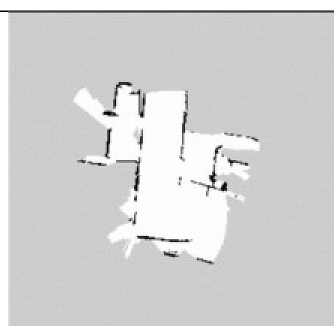
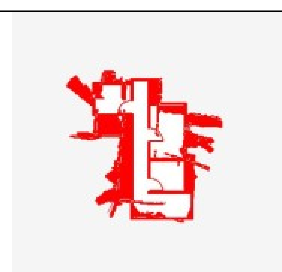
L:ampiran 5 Launch File SLAM

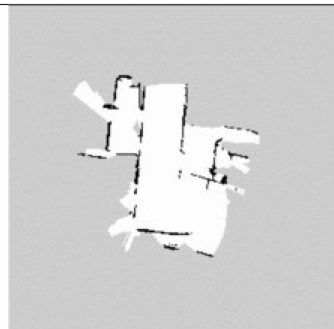
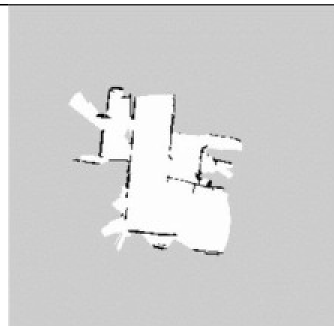
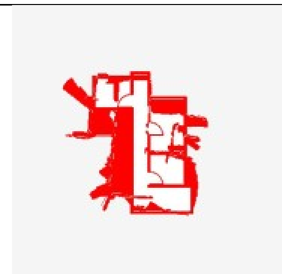
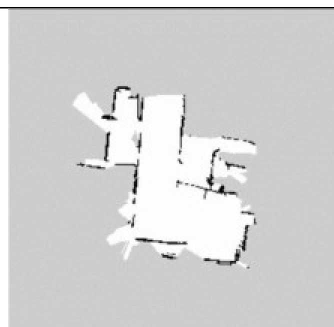
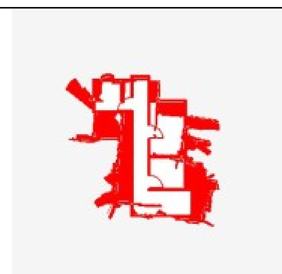
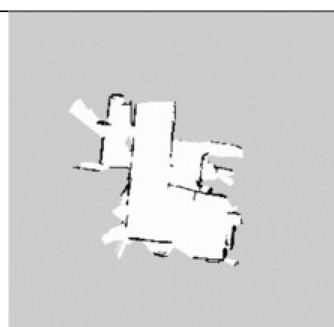
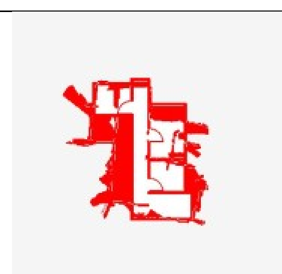
```
<?xml version="1.0"?>
<launch>
  <!-- kinect -->
  <include file="$(find freenect_launch)/launch/freenect.launch">
    <arg name="depth_registration" value="true" />
  </include>

  <!-- Kinect cloud to laser scan -->
    <node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan"
name="depthimage_to_laserscan">
      <remap from="image"       to="/camera/depth_registered/image_raw"/>
      <remap from="camera_info" to="/camera/depth_registered/camera_info"/>
      <remap from="scan"        to="/kinect_scan"/>
      <param name="range_max" type="double" value="4"/>
    </node>

  <!-- another-config -->
  <arg name="model" default="$(find
autonomus_mobile_robot)/urdf/mobile_robot.urdf.xacro"/>
  <param name="robot_description" command="$(find xacro)/xacro $(arg model)" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" >
  </node>
  <node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui"
type="joint_state_publisher_gui" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
autonomus_mobile_robot)/config/amcl_nav.rviz"/>
  <node pkg="tf" type="static_transform_publisher"
name="image_to_scan_broadcaster" args="0 0 0 0 0 0 \/camera_depth_frame camera
100" />
```

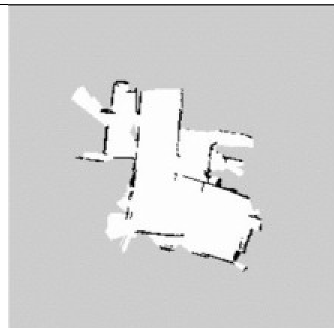
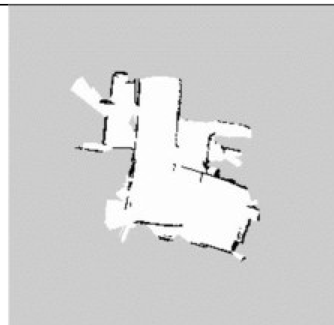
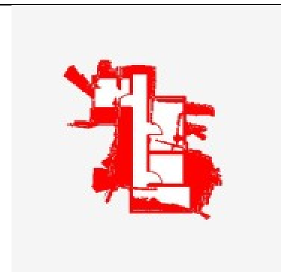
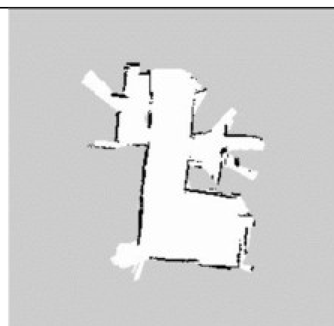
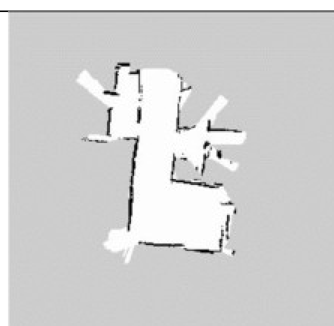
```
<arg name="pi/2" value="1.5707963267948966" />

<arg name="pi" value="3.14" />

<node pkg="tf" type="static_transform_publisher" name="camera_optical" args="-0 0
0.83 0 0 0) /base_link /camera_link 100" />


<!-- Odometry -->

<node pkg="nodelet" type="nodelet" name="rgbd_sync" args="standalone
rtabmap_sync/rgbd_sync" output="screen">

  <remap from="rgb/image"       to="/camera/rgb/image_rect_color"/>

  <remap from="depth/image"     to="/camera/depth_registered/image_raw"/>

  <remap from="rgb/camera_info"  to="/camera/rgb/camera_info"/>

  <remap from="rgbd_image"      to="rgbd_image"/> <!-- output -->


    <!-- Should be true for not synchronized camera topics

         (e.g., false for kinectv2, zed, realsense, true for xtion, kinect360)-->

  <param name="approx_sync"      value="true"/>

</node>

<node pkg="rtabmap_odom" type="rgbd_odometry" name="rgbd_odometry"
output="screen">

  <param name="subscribe_rgbd" type="bool"   value="true"/>

  <param name="frame_id"       type="string" value="base_link"/> <!-- ubah jadi
"base_link" jika tidak menggunakan fusion -->

  <remap from="rgbd_image" to="rgbd_image"/>

  <remap from="odom" to="vo"/> <!-- Mengubah topik odom menjadi "odom_rgbd"
jika mengaktifkan fusion -->

</node>

<node pkg="my_robot_pkg" type="ekf_odom_pub" name="ekf_odom_pub">

</node>

<node pkg="my_robot_pkg" type="rviz_click_to_2d" name="rviz_click_to_2d">

</node>

<node pkg="rosserial_python" type="serial_node.py" name="serial_node">

  <param name="port" value="/dev/ttyUSB0"/>
```

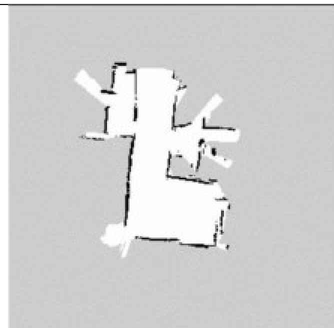
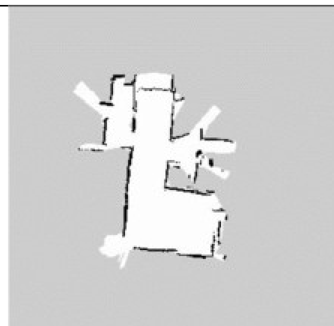
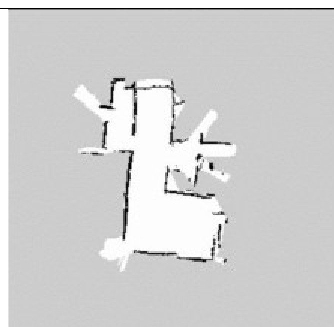
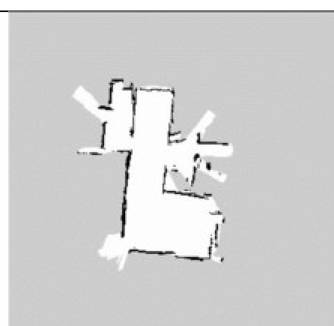
```
    <param name="baud" value="115200"/>

  </node>

  <node pkg="tf" type="static_transform_publisher" name="imu_broadcaster" args="0
0.06 0.02 0 0 0 base_link imu_link 30" />

  <node pkg="tf" type="static_transform_publisher" name="base_link_broadcaster"
args="0 0 0.09 0 0 0 base_footprint base_link 30" />

  <remap from="odom" to="odom_data_quat" />

  <remap from="imu_data" to="imu_data" />

  <node pkg="robot_pose_ekf" type="robot_pose_ekf" name="robot_pose_ekf">

    <param name="output_frame" value="odom"/>

    <param name="base_footprint_frame" value="base_footprint"/>

    <param name="freq" value="30.0"/>

    <param name="sensor_timeout" value="1.0"/>

    <param name="odom_used" value="true"/>

    <param name="imu_used" value="false"/>

    <param name="vo_used" value="true"/>

    <param name="gps_used" value="false"/>

    <param name="debug" value="false"/>

    <param name="self_diagnose" value="false"/>

  </node>


  <!-- SLAM -->

 <node pkg="gmapping" type="slam_gmapping" name="gmapping_thing"
output="screen" >

   <remap from="scan" to="/kinect_scan" />

   <param name="odom_frame" value="/odom" />

   <param name="base_frame" value="/base_link" />

 </node>


</launch>
```

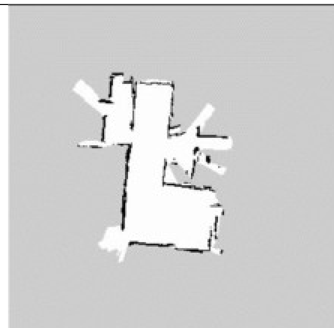
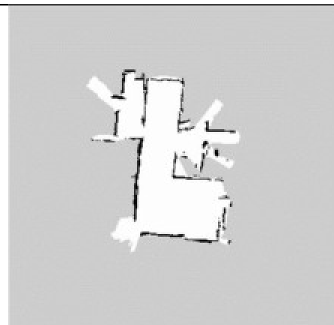
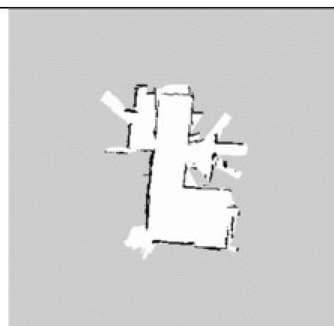
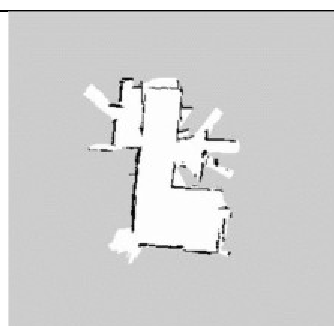
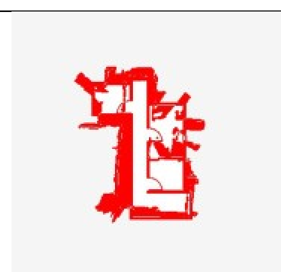
Lampiran 6 Data Hasil Pemetaan Ruang Uji 1 (Laboratorium Sistem Kendali dan instrumentasi)

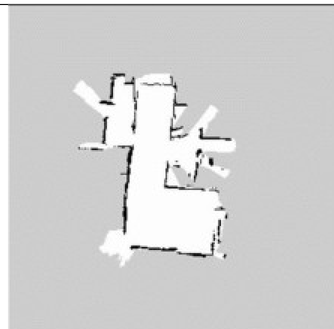
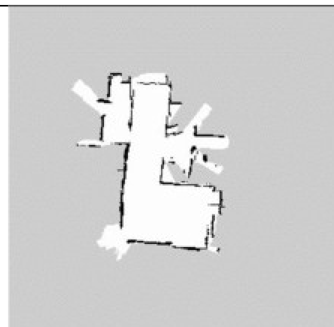
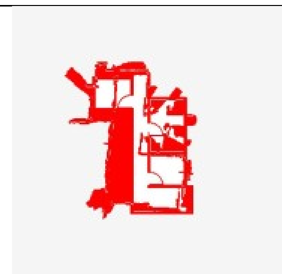
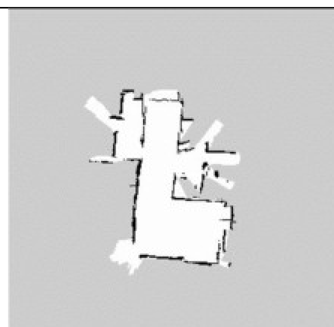
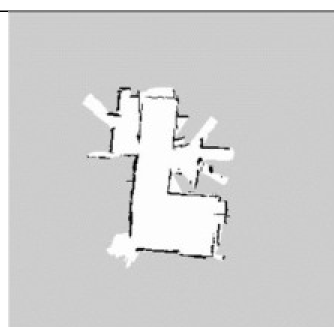
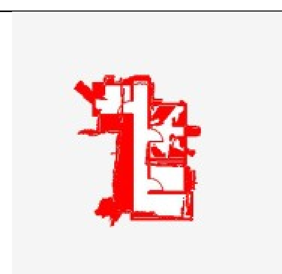
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
1			7.05%
2			7.94%
3			9.18%
4			9.26%

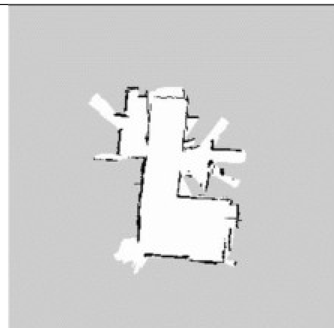
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
5			9.26%
6			9.46%
7			10.57%
8			9.83%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
9			11.00%
10			11.17%
11			12.12%
12			10.87%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
13			10.19%
14			10.11%
15			9.90%
16			9.68%

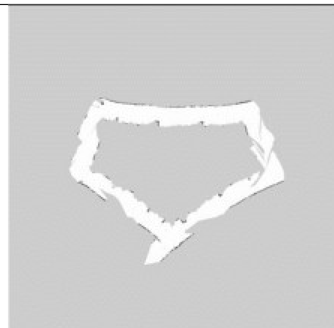
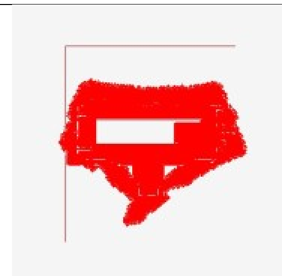
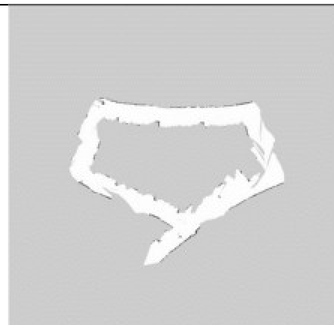
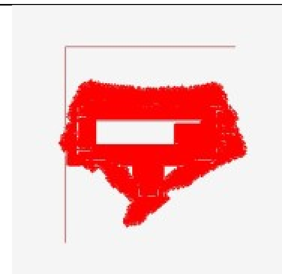
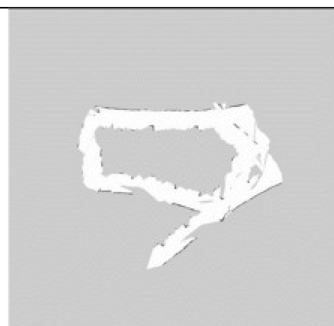
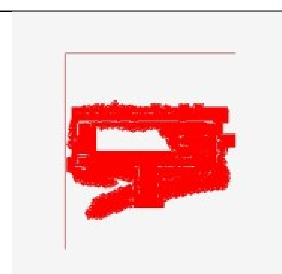
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
17			9.67%
18			9.05%
19			8.01%
20			9.20%

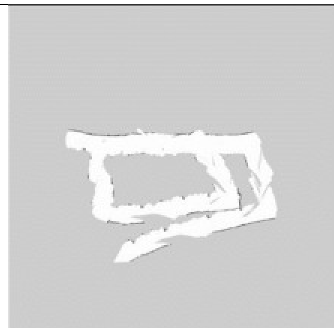
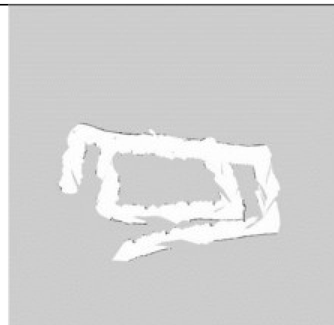
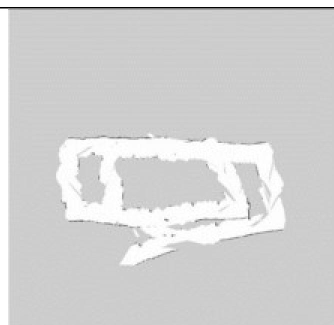
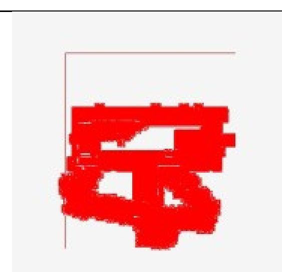
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
21			9.96%
22			10.37%
23			9.06%
24			8.79%

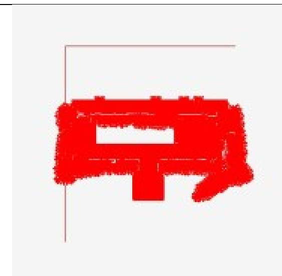
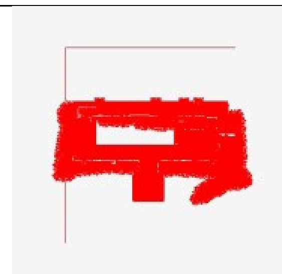
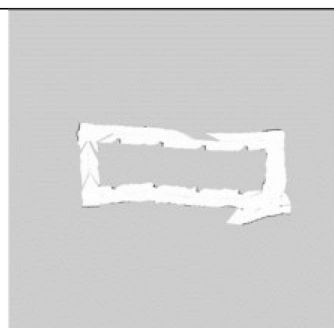
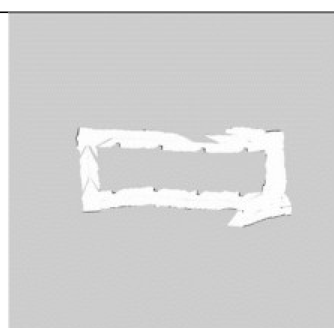
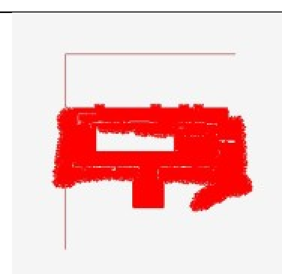
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
25			9.26%
Rata-rata			9,24%

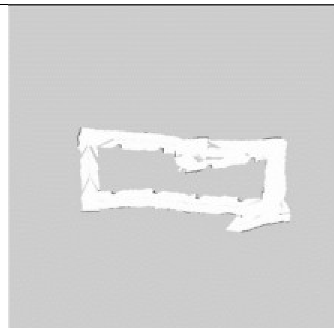
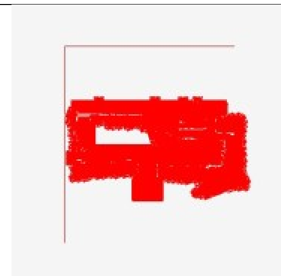
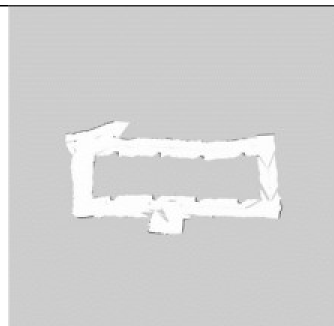
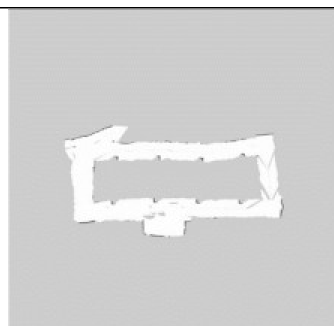
Lampiran 7 Data Hasil Pemetaan Ruang Uji 2 (Gedung Center Of Technology Lantai 2)

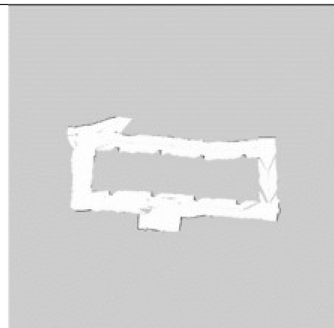
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
1			17.02%
2			18.76%
3			19.34%
4			17.64%

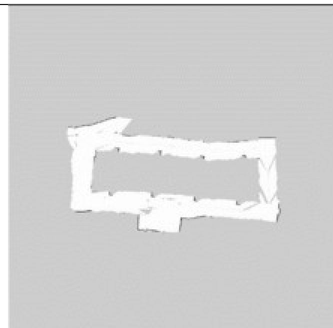
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
5			19.63%
6			19.63%
7			18.32%
8			17.56%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
9			20.62%
10			22.40%
11			23.43%
12			22.72%

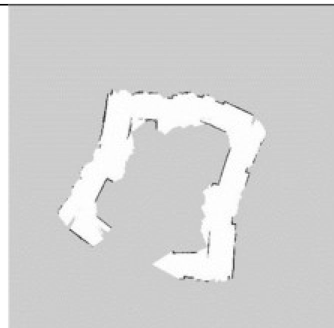
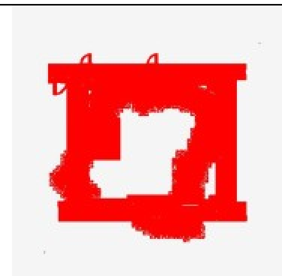
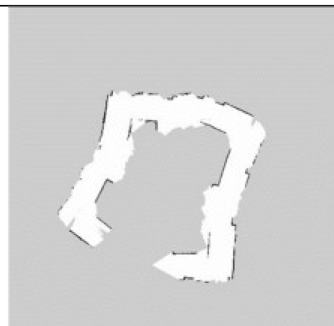
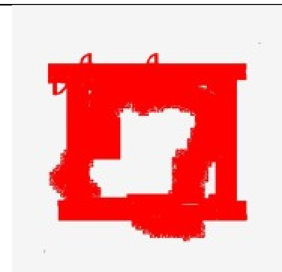
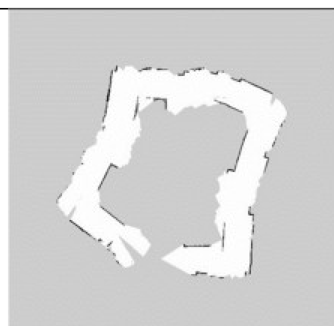
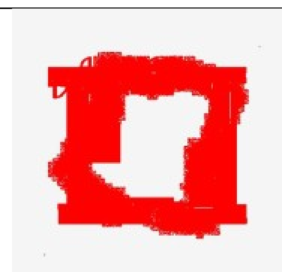
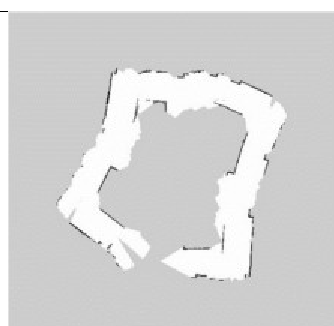
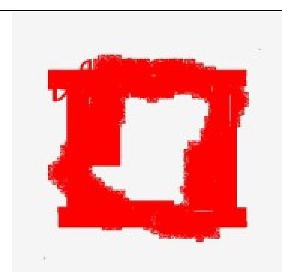
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
13			19.58%
14			19.38%
15			19.38%
16			18.42%

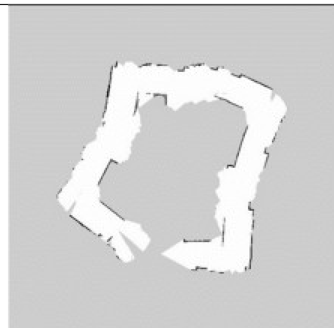
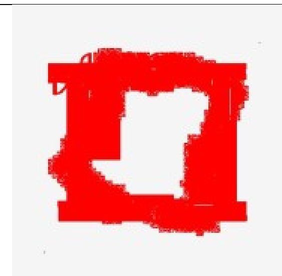
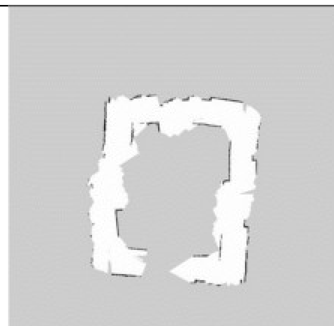
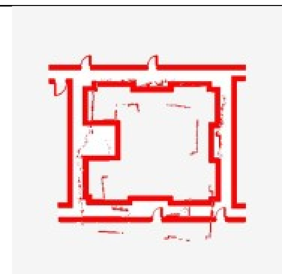
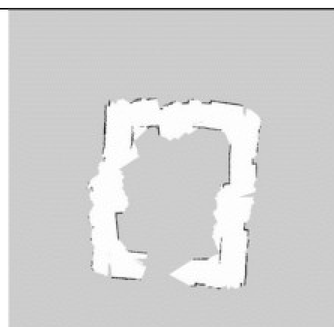
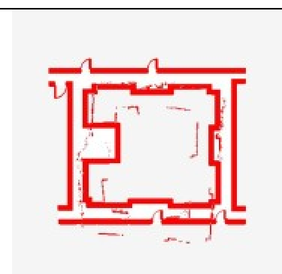
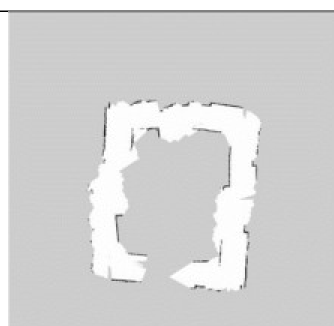
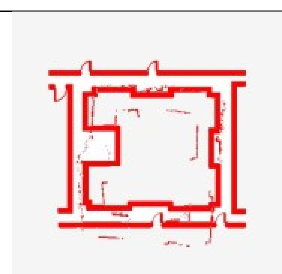
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
17			19.64%
18			17.55%
19			15.51%
20			15.51%

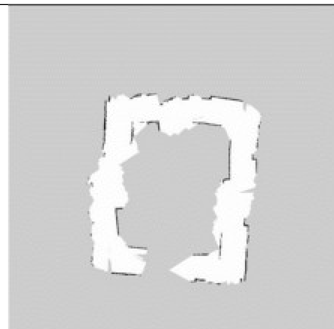
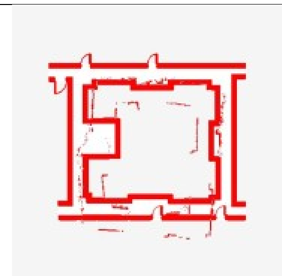
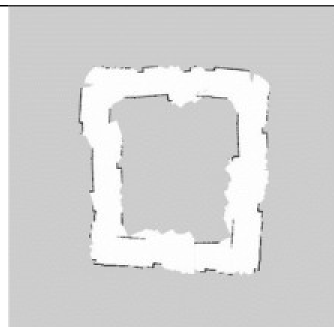
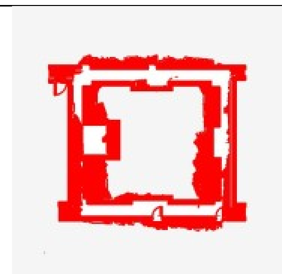
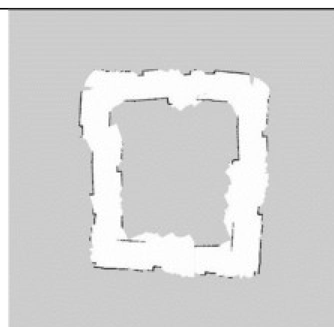
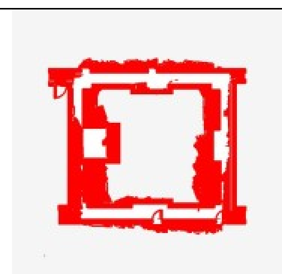
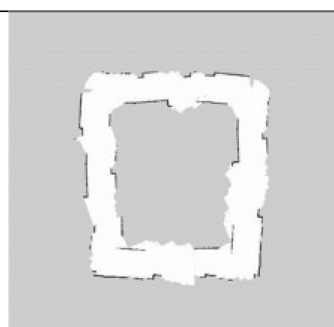
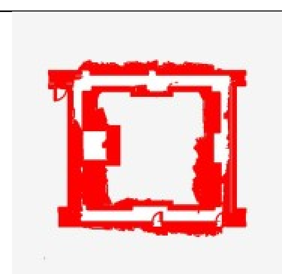
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
21			15.32%
22			15.14%
23			15.32%
24			15.14%

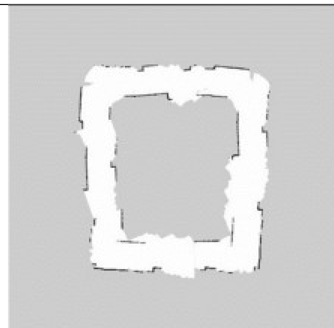
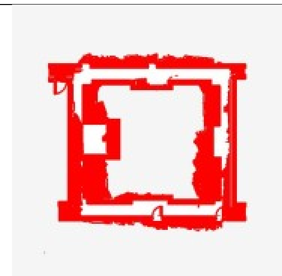
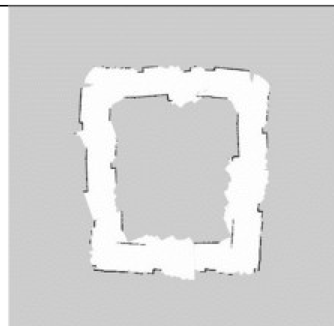
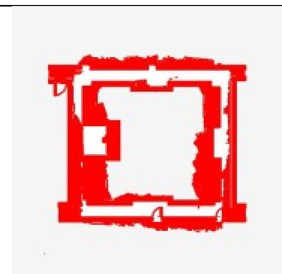
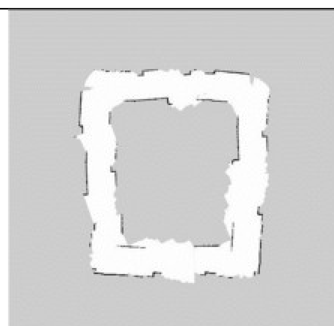
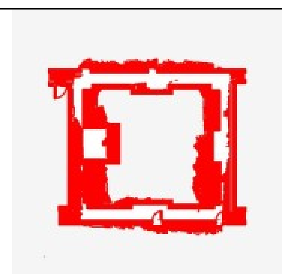
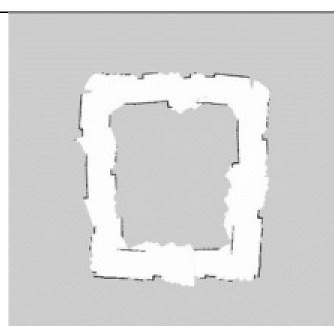
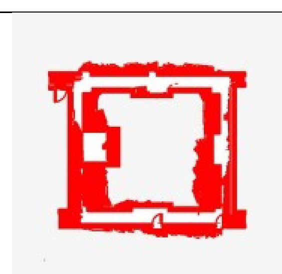
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
25			15.11%
Rata-rata			18,32%

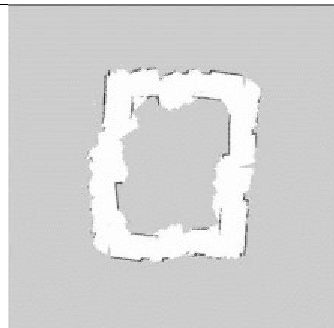
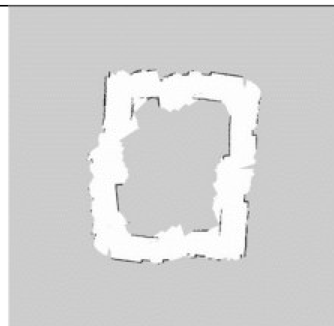
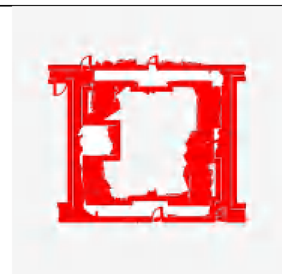
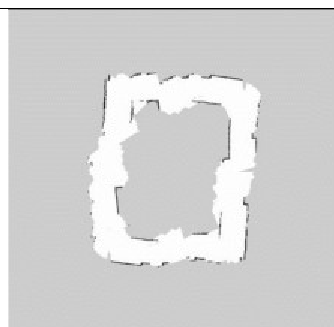
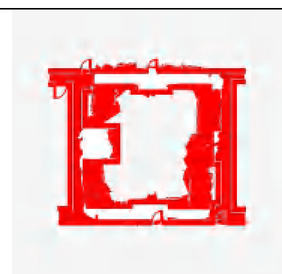
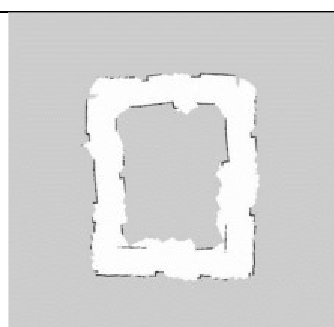
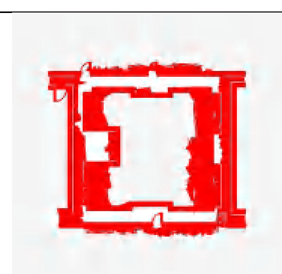
Lampiran 8 Data Hasil Pemetaan Ruang Uji 3 (Gedung Elektro Lantai 3)

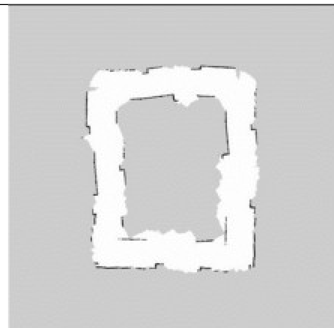
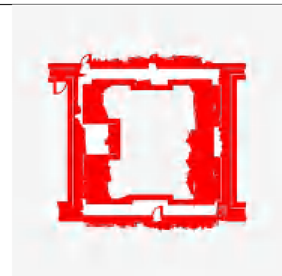
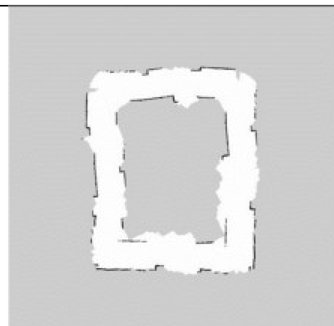
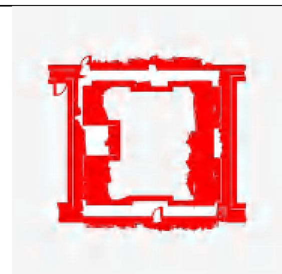
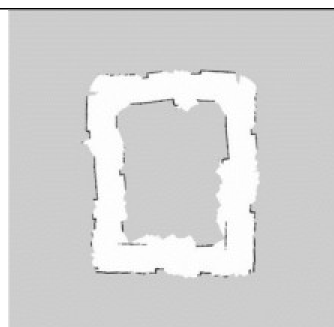
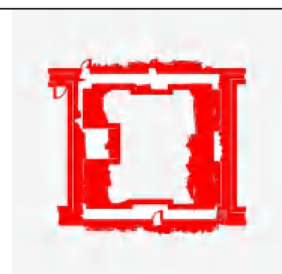
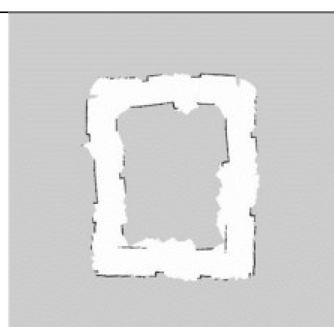
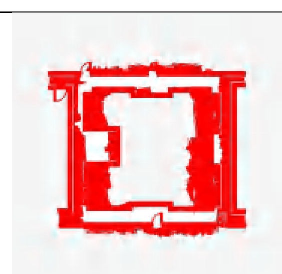
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
1			30.35%
2			30.54%
3			29.87%
4			29.66%

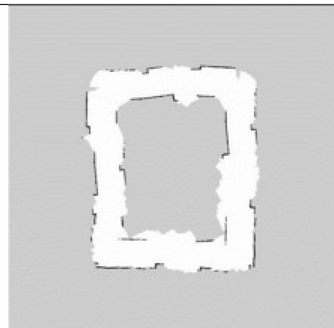
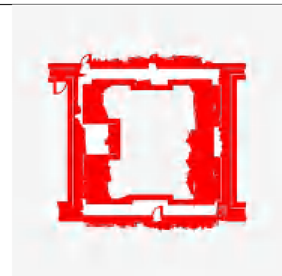
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
5			29.39%
6			19.41%
7			19.41%
8			19.41%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
9			19.41%
10			19.23%
11			19.23%
12			19.23%

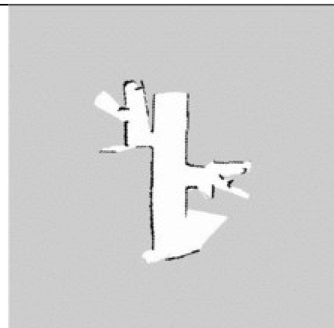
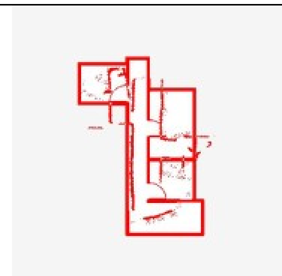
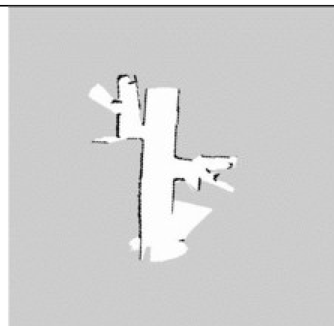
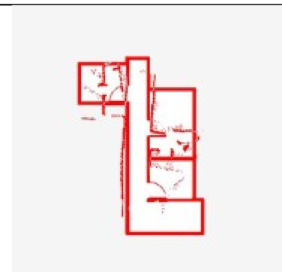
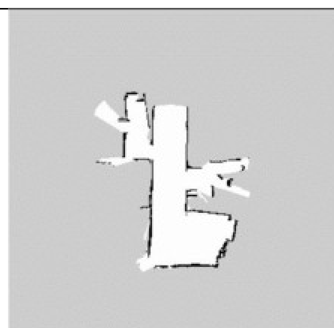
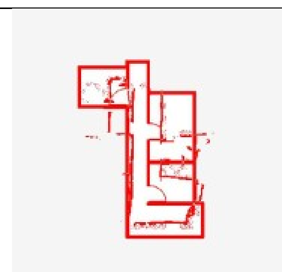
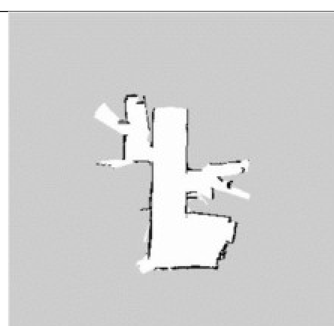
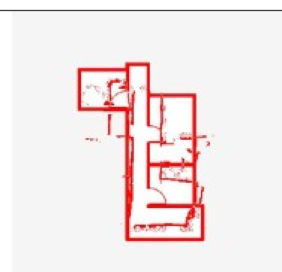
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
13			19.23%
14			19.23%
15			19.23%
16			19.23%

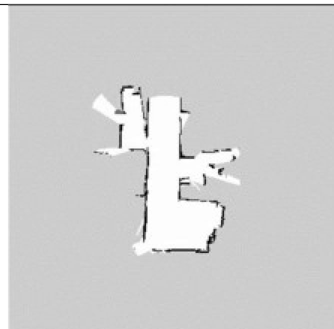
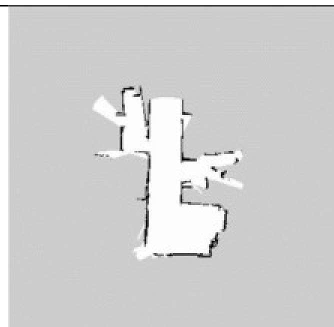
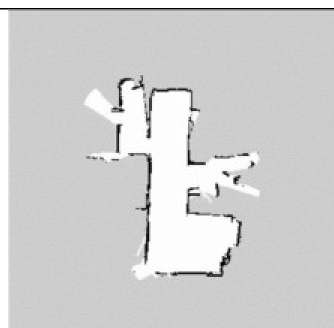
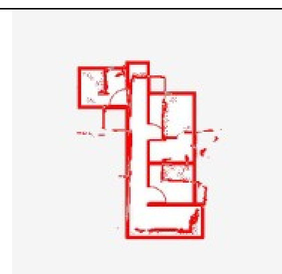
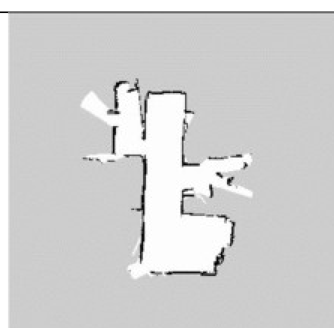
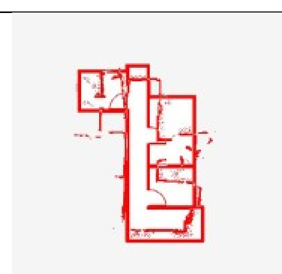
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
17			20.10%
18			20.10%
19			20.10%
20			20.64%

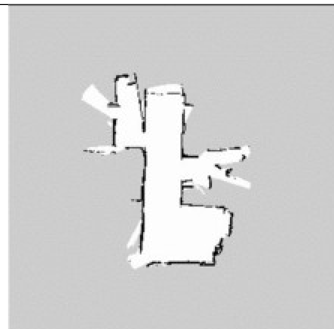
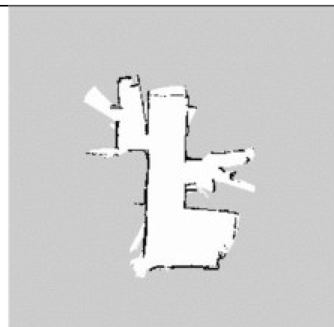
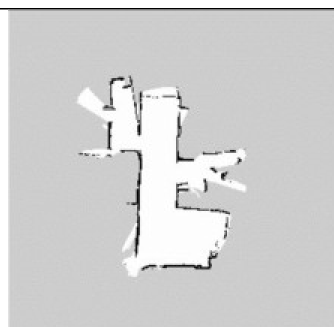
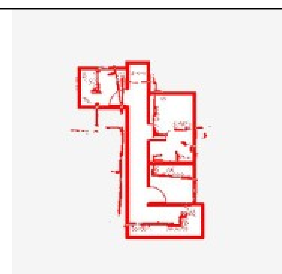
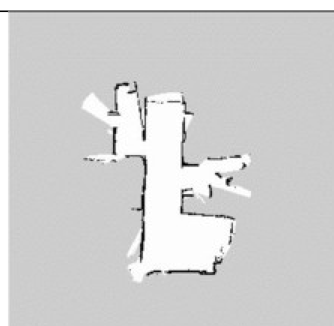
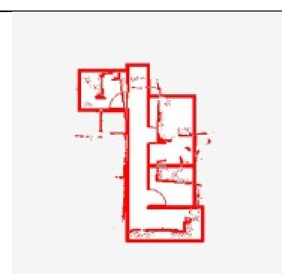
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
21			20.64%
22			20.64%
23			20.64%
24			20.43%

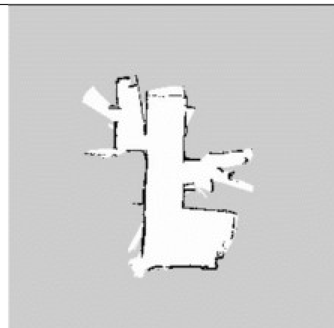
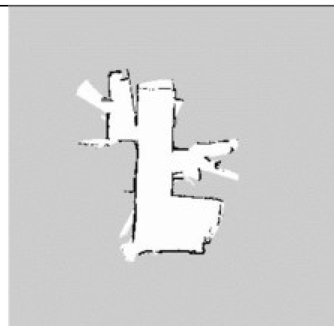
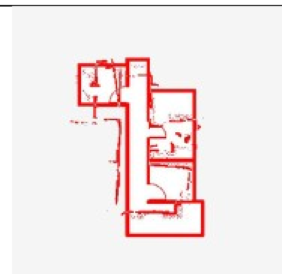
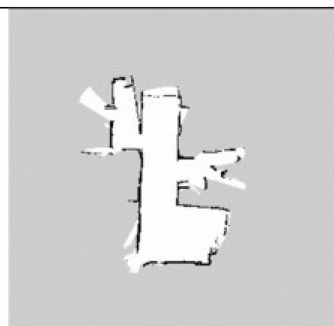
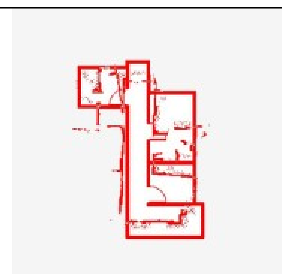
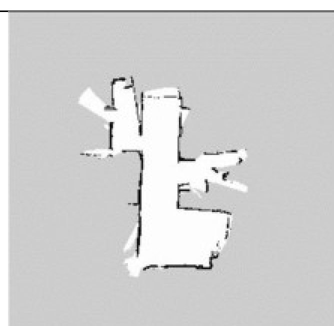
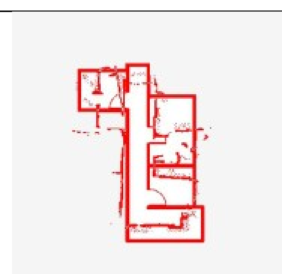
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
25			20.43%
Rata-rata			21,83%

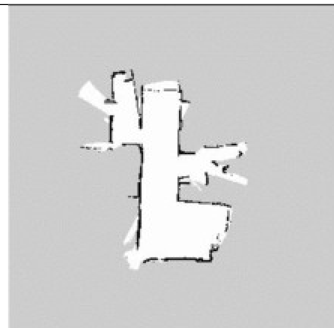
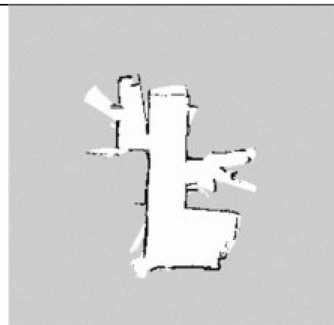
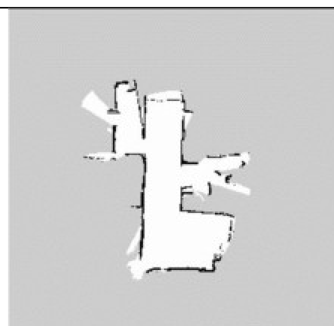
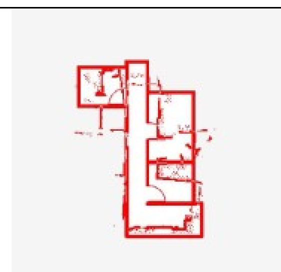
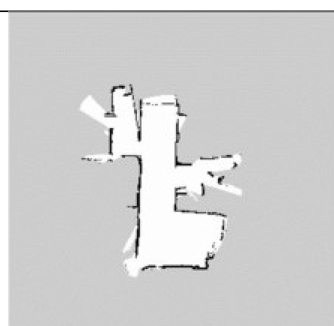
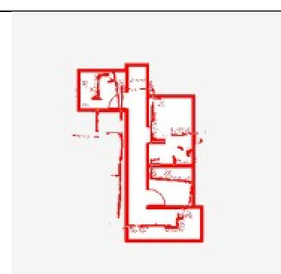
Lampiran 9 Data Hasil Pemetaan pada kondisi luminensi skenario 1

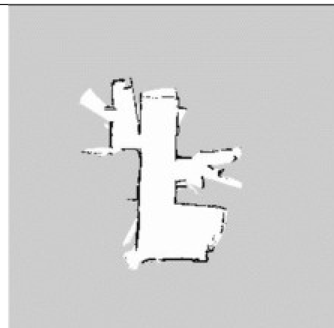
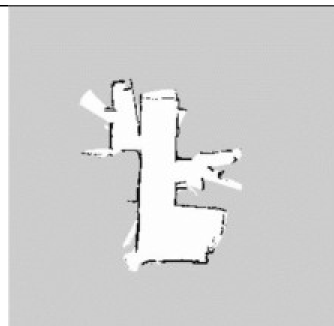
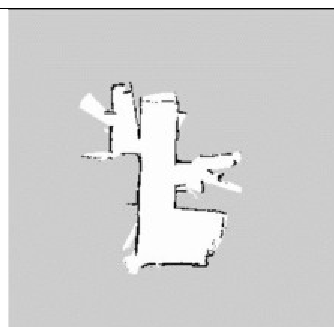
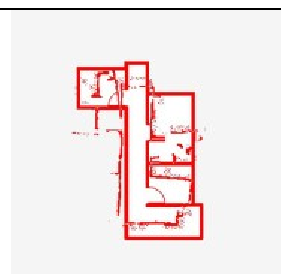
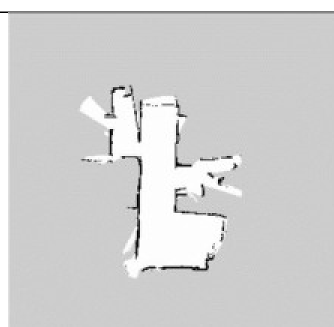
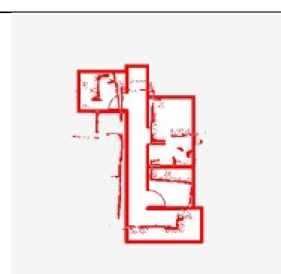
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
1			5.17%
2			4.90%
3			5.54%
4			5.54%

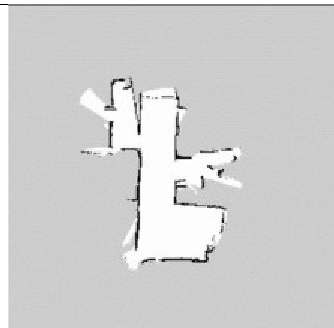
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
5			5.08%
6			5.11%
7			5.71%
8			5.75%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
9			5.61%
10			5.36%
11			5.75%
12			5.57%

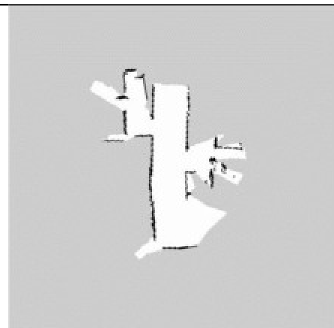
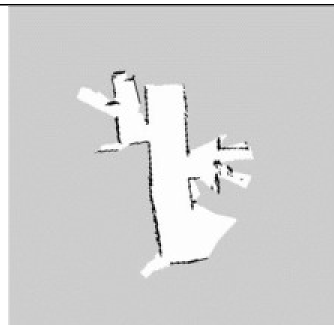
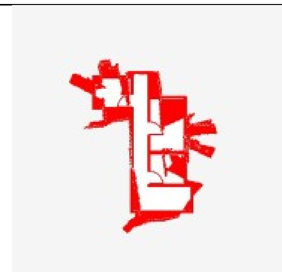
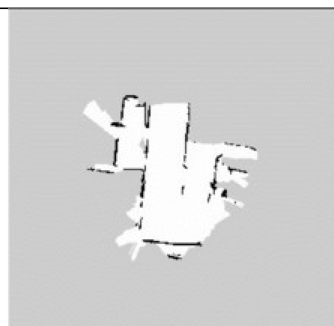
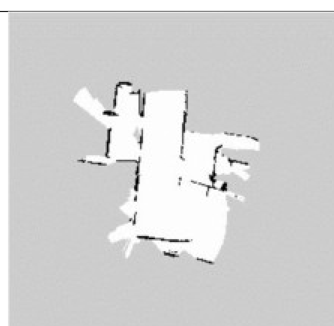
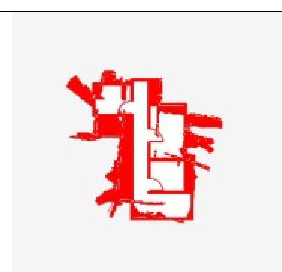
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
13			5.49%
14			5.62%
15			5.71%
16			5.79%

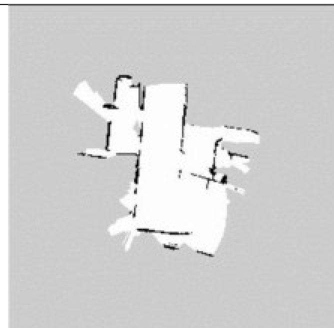
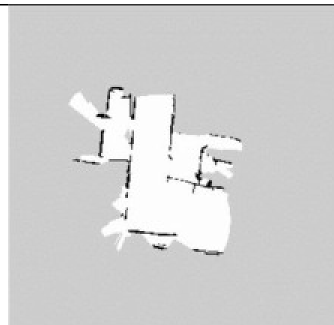
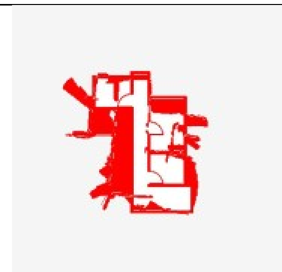
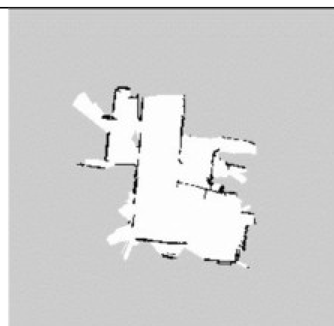
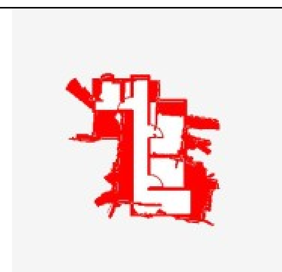
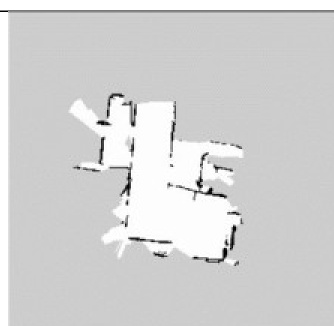
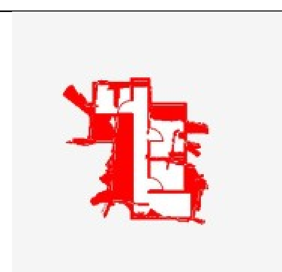
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
17			5.79%
18			5.34%
19			5.58%
20			5.72%

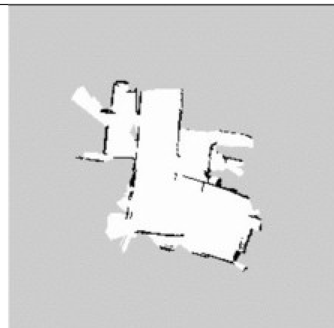
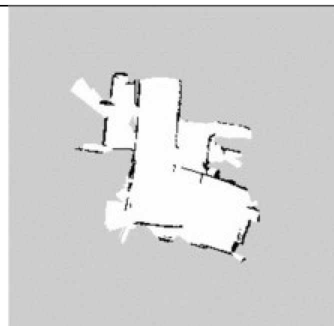
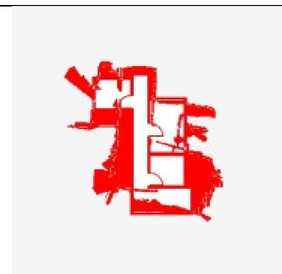
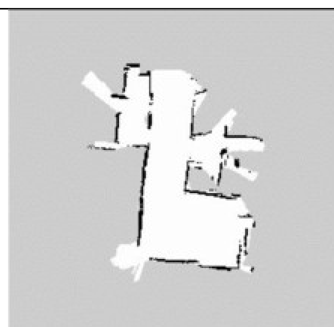
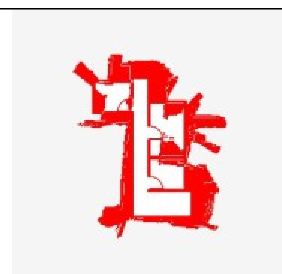
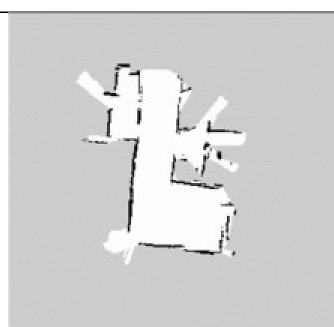
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
21			5.72%
22			5.72%
23			5.72%
24			5.72%

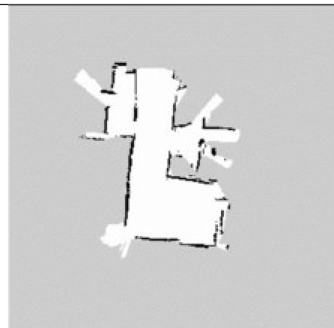
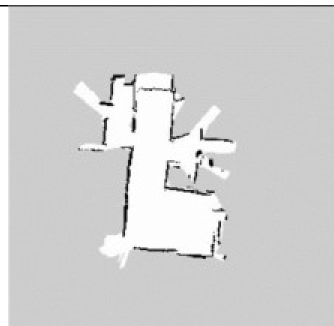
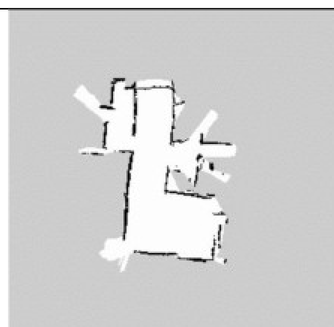
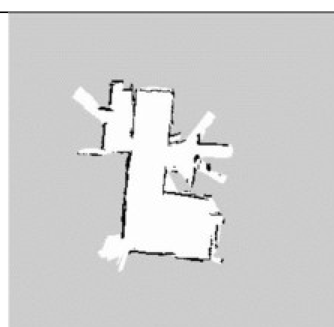
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
25			5.72%
Rata-rata			5,55%

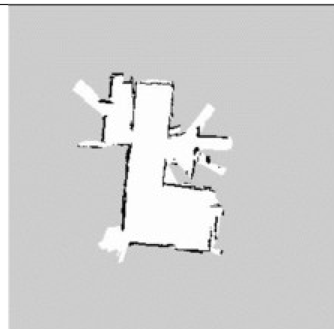
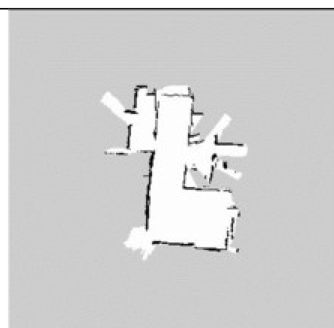
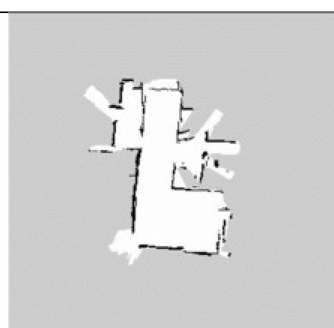
Lampiran 10 Data Hasil Pemetaan pada kondisi luminensi skenario 2

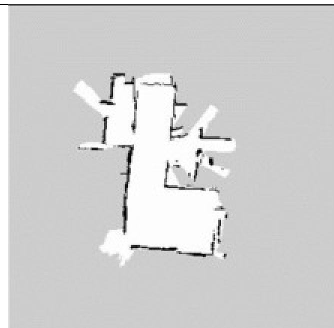
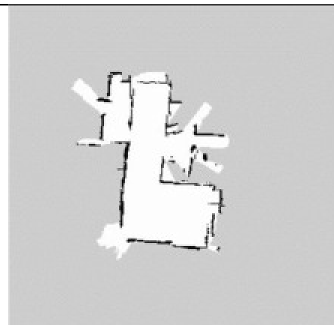
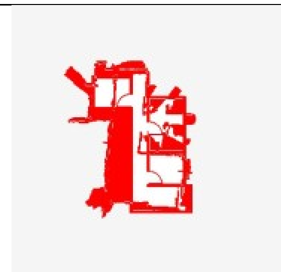
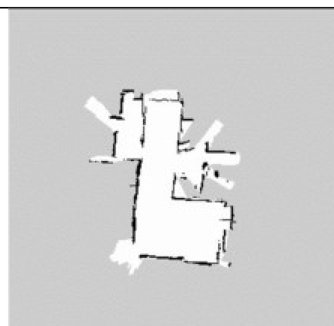
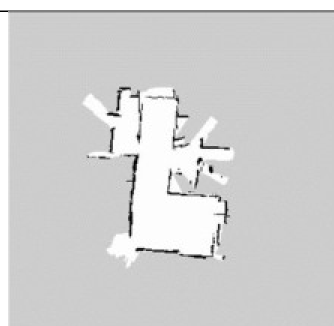
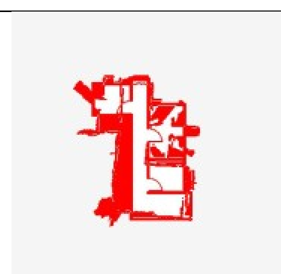
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
1			7.05%
2			7.94%
3			9.18%
4			9.26%

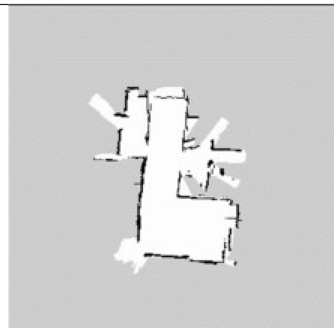
Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
5			9.26%
6			9.46%
7			10.57%
8			9.83%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
9			11.00%
10			11.17%
11			12.12%
12			10.87%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
13			10.19%
14			10.11%
15			9.90%
16			9.68%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
17			9.67%
18			9.05%
19			8.01%
20			9.20%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
21			9.96%
22			10.37%
23			9.06%
24			8.79%

Percobaan ke-n	Hasil pemetaan	Komparasi	% Error (RMSE)
25			9.26%
Rata-rata			9,24%