menggunakan 2 *node* sensor *gateway* menerima data sebanyak 606 data, dengan menggunakan 3 *node* sensor *gateway* menerima data sebanyak 607, secara keseluruhan sistem tanpa *multitasking* menerima data sebanyak 85.78% dari total data yang dikirim. Sedangkan pada sistem dengan *multitasking*, *gateway* dengan pengujian menggunakan 1 *node* sensor menerima data sebanyak 303 data, dengan menggunakan 2 *node* sensor *gateway* menerima data sebanyak 607 data, dengan menggunakan 3 *node* sensor *gateway* menerima data sebanyak 821 data, secara keseluruhan sistem dengan *multitasking* data yang diterima oleh *gateway* meningkat menjadi 98.49% dari total data yang dikirim. Dengan penggunaan *multitasking* pada *gateway* memungkinkan penerimaan dan pemrosesan data yang lebih efisien, dengan jumlah data yang lebih tinggi dalam waktu yang lebih singkat. *Multitasking* meningkatkan kinerja *gateway* dan memungkinkan sistem untuk menangani beban data yang lebih besar secara efektif.

5.2 SARAN

Sehubung dengan penyelesain penulisan skripsi ini, penulis bermaksud untuk menyampaikan beberapa saran terhadap pengembangan sistem, antara lain:

1. Sistem ini dapat diterapkan tidak hanya pada kasus hidroponik, tetapi juga dapat diterapkan pada berbagai jenis sistem yang memiliki tingkat kompleksitas yang berbeda.
2. Sistem yang telah dibuat masih dapat dikembangkan dan dioptimalkan dengan mengimplementasikan lebih banyak *API Reference* yang dimiliki oleh *freeRTOS* seperti fungsi *xSemaphore*.

DAFTAR PUSTAKA

H. A., & Aldossary, M. (2021). Energy-Efficient Edge-Fog-Cloud
nitecture for IoT-Based Smart Agriculture Environment. *IEEE Access*, *9*,
480–110492. https://doi.org/10.1109/ACCESS.2021.3101397

Chaudhari, B. S., Zennaro, M., & Borkar, S. (2020). LPWAN technologies: Emerging application characteristics, requirements, and design considerations. *Future Internet*, *12*(3). https://doi.org/10.3390/fi12030046

Cohen, L., Manion, L., & Morrison, K. (2020). Quantitative data analysis. *Research Methods in Education*, 519–576. https://doi.org/10.4324/9780203029053-36

Datta, S. K., & Bonnet, C. (2017). An edge computing architecture integrating virtual IoT devices. *2017 IEEE 6th Global Conference on Consumer Electronics, GCCE 2017*, *2017-Janua*, 1–3. https://doi.org/10.1109/GCCE.2017.8229253

Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., & Hindia, M. N. (2018). An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet of Things Journal*, *5*(5), 3758–3773. https://doi.org/10.1109/JIOT.2018.2844296

Frima Yudha, P. S., & Sani, R. A. (2019). Implementasi Sensor Ultrasonik Hc-Sr04 Sebagai Sensor Parkir Mobil Berbasis Arduino. *EINSTEIN E-JOURNAL*, *5*(3). https://doi.org/10.24114/einstein.v5i3.12002

Hardana, Radian Ferrari Isputra. (2019). *Membuat Aplikasi IoT: INTERNET OF THINGS*. Yogyakarta: Lokomedia.

Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I., & Imran, M. (2018). The Role of Edge Computing in Internet of Things. *IEEE Communications Magazine*, *56*(11), 110–115. https://doi.org/10.1109/MCOM.2018.1700906

Inam, R., Mäki-turja, J., Sjödin, M., Ashjaei, S. M. H., & Afshar, S. (2011). Hierarchical Scheduling Framework Implementation in FreeRTOS. *Proceedings of the 16th Conference on Emerging Technologies & Factory Automation (ETFA)*.

[illegible]. A., Salih, A. A., Al-zebari, A., Omar, N., Hasan, S. S., Kak, S. F., & [illegible]him, I. M. (2021). *Scheduling Algorithms Implementation for Real Time*

Operating Systems : A Review. *September*. https://doi.org/10.9734/ajrcos/2021/v11i430269

Istiana, T., Mardyansyah, R. Y., & Dharmawan, G. . B. (2020). Kajian Pemanfaatan IoT Berbasis LPWAN Untuk Jaringan Akuisisi Data ARG. *Elektron : Jurnal Ilmiah*, *12*(1), 1–6. https://doi.org/10.30630/eji.12.1.155

Jayanti, H. Y. (2018). Peramalan Pendapatan Reksa Dana Dalam Setahun Menggunakan Metode Regresi Linier Sederhana. *Jurnal FIKI(Jurnal Teknologi Informasi Dan Komunikasi)*, *VIII*(2), 136–139. http://jurnal.unnur.ac.id/index.php/jurnalfiki

Kavitha, S., Varuna, S., & Ramya, R. (2017). A comparative analysis on linear regression and support vector regression. *Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016*. https://doi.org/10.1109/GET.2016.7916627

Kurniawan, R., Setiawan, I., & Sumardi. (2010). *Multitasking* Pada Mikrokontroler Atmega16 Menggunakan Real Time Operating System (Rtos) Jenis Cooperative. *Makalah Seminar Tugas Akhir*.

Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, *5*(1), 1–7. https://doi.org/10.1016/j.icte.2017.12.005

Patel, K., & Keyur. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *Universidad Iberoamericana Ciudad de México*, *May*, 6123,6131. http://www.opjstamnar.com/download/Worksheet/Day-110/IP-XI.pdf

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

& Dustdar, S. (2016). *CLOUD COVER WHY DO WE NEED EDGE MPUTING? The Promise of Edge Computing*. *0018*.

Thamrin, R. Z., Samijayani, O. N., Rahmatia, S., Adrianto, D., & Enriko, I. K. A. (2020). Implementation of LoRa End-Device in Sensor Network System for Indoor Application. *2020 IEEE International Conference on Communication, Networks and Satellite, Comnetsat 2020 - Proceedings*, 208–212. https://doi.org/10.1109/Comnetsat50391.2020.9329003

Wisnu Jatmiko, Petrus Mursanto, D. (2015). Real Time Operating System (RTOS). *Real Time Operating System (RTOS) : Teori Dan Aplikasi*, 101–118.

LAMPIRAN

Lampiran 1 *Source Code Gateway* Tanpa *Multitasking*

```
// Program Name : Implementation of Multitasking in Edge Computing for Hydroponic
Plant Monitoring Farm
// Contributor in This Program:
// 1. Conceptualize, Idea: Adnan, S.T., M.T., Ph.D.
// 2. Desain: Adnan, S.T., M.T., Ph.D. and Devy Noviani Badjarad
// 3. Implementation: Devy Noviani Badjarad
// 4. Testing: Devy Noviani Badjarad
// 5. Supervision: Adnan, S.T., M.T., Ph.D. and Dr. Eng. Zulkifli Tahir, S.T.,
M.Sc.

#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <WiFiMulti.h>
#include "SX1272.h"
#include "ThingSpeak.h"
#include <CircularBuffer.h>
#include <ESP32Time.h>
#include <string.h>

#define TS_ENABLE_SSL

#ifdef ARDUINO
// IMPORTANT when using an Arduino only. For a Raspberry-based gateway the
distribution uses a radio.makefile file
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////
// please uncomment only 1 choice
//
// uncomment if your radio is an HopeRF RFM92W, HopeRF RFM95W, Modtronix inAir9B,
NiceRF1276
// or you known from the circuit diagram that output use the PABOOST line instead
of the RFO line
#define PABOOST
//////////////////////////////////////////////////////////////////////////////
#endif
// IMPORTANT
//////////////////////////////////////////////////////////////////////////////
// please uncomment only 1 choice
//#define BAND868
#define BAND900
#define BAND433
```

```
// For a Raspberry-based gateway the distribution uses a radio.makefile file that
can define MAX_DBM
//
#ifndef MAX_DBM
#define MAX_DBM 14
#endif

#ifndef LORA_PREAMBLE_LENGTH
#define LORA_PREAMBLE_LENGTH 8
#endif

#ifndef ARDUINO
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#ifdef GETOPT_ISSUE
int getopt(int argc, char *const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
#endif
#include <getopt.h>
#include <termios.h>
#include <signal.h>
#include <sys/time.h>
#include <time.h>
#include <math.h>
#include <string.h>
#endif

#ifdef ARDUINO
// and SPI library on Arduino platforms
#include <SPI.h>

#define PRINTLN Serial.println("")
#define PRINT_CSTSTR(fmt, param) Serial.print(F(param))
#define PRINT_STR(fmt, param) Serial.print(param)
#define PRINT_VALUE(fmt, param) Serial.print(param)
#define PRINT_HEX(fmt, param) Serial.print(param, HEX)
#define FLUSHOUTPUT Serial.flush();
#else
#define PRINTLN printf("\n")
#define PRINT_CSTSTR(fmt, param) printf(fmt, param)
#define PRINT_STR(fmt, param) PRINT_CSTSTR(fmt, param)
#define PRINT_VALUE(fmt, param) PRINT_CSTSTR(fmt, param)
RINT_HEX(fmt, param) PRINT_VALUE(fmt, param)
LUSHOUTPUT fflush(stdout);


3UG
```

```
#define DEBUGLN PRINTLN
#define DEBUG_CSTSTR(fmt, param) PRINT_CSTSTR(fmt, param)
#define DEBUG_STR(fmt, param) PRINT_CSTSTR(fmt, param)
#define DEBUG_VALUE(fmt, param) PRINT_VALUE(fmt, param)
#else
#define DEBUGLN
#define DEBUG_CSTSTR(fmt, param)
#define DEBUG_STR(fmt, param)
#define DEBUG_VALUE(fmt, param)
#endif

void setup() {
  int e;
  #ifdef ARDUINO
  delay(3000);
  randomSeed(analogRead(14));

  // Open serial communications and wait for port to open:
  #ifdef __SAMD21G18A__
  SerialUSB.begin(38400);
  #else
  Serial.begin(38400);
  #endif

  //CONNECTED TO WIFI
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  while (wifiMulti.run() != WL_CONNECTED)
  {
    delay(1000);
  }
  Serial.println("Connected");
  Serial.println(WiFi.localIP());

  ThingSpeak.begin(client); // Initialize ThingSpeak


  #if defined ARDUINO && defined SHOW_FREEMEMORY && not defined __MK20DX256__ && not
defined   __MKL26Z64__   &&   not   defined   __SAMD21G18A__   &&   not   defined
_VARIANT_ARDUINO_DUE_X_
    // Print a start message
    Serial.print(freeMemory());
    Serial.println(F(" bytes of free memory."));
  #endif

    (time(NULL));
```

```
// Power ON the module
e = sx1272.ON();

PRINT_CSTSTR("%s", "^$**********Power ON: state ");
PRINT_VALUE("%d", e);
PRINTLN;

if (!e)
{
  radioON = true;
  startConfig();
}

e = sx1272.getSyncWord();

if (!e)
{
  PRINT_CSTSTR("%s", "^$Default sync word: 0x");
  PRINT_HEX("%X", sx1272._syncWord);
  PRINTLN;
}

if (optSW != 0x12)
{
  e = sx1272.setSyncWord(optSW);

  PRINT_CSTSTR("%s", "^$Set sync word to 0x");
  PRINT_HEX("%X", optSW);
  PRINTLN;
  PRINT_CSTSTR("%s", "^$LoRa sync word: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;
}

FLUSHOUTPUT;
delay(1000);

#ifdef LORA_LAS
loraLAS.ON(LAS_ON_WRESET);
#endif

#ifdef CAD_TEST
PRINT_CSTSTR("%s", "Do CAD test\n");
#endif
```

```
 defined ARDUINO && defined DOWNLINK

nlinkCheckTime = millis();
```

```
  #endif

}

// we could use the CarrierSense function added in the SX1272 library, but it is
more convenient to duplicate it here
// so that we could easily modify it for testing
//
// in v1.5 the "only once" behavior is implemented for the gateway when it transmit
downlink packets
// to avoid blocking the gateway on a busy channel. Therefore from v1.5 the
implementation differs from the
// carrier sense function added in the SX1272 library
//
int CarrierSense(bool onlyOnce = false) {

  int e;
  bool carrierSenseRetry = false;

  if (send_cad_number)
  {
    do
    {
      do
      {

        // check for free channel (SIFS/DIFS)
        startDoCad = millis();
        e = sx1272.doCAD(send_cad_number);
        endDoCad = millis();

        PRINT_CSTSTR("%s", "--> CAD duration ");
        PRINT_VALUE("%ld", endDoCad - startDoCad);
        PRINTLN;

        if (!e)
        {
          PRINT_CSTSTR("%s", "OK1\n");

          if (extendedIFS)
          {
            // wait for random number of CAD
            #ifdef ARDUINO
              uint8_t w = random(1, 8);
            #else
              uint8_t w = rand() % 8 + 1;
            #endif

            PRINT_CSTSTR("%s", "--> waiting for ");
```

```
      PRINT_VALUE("%d", w);
      PRINT_CSTSTR("%s", " CAD = ");
      PRINT_VALUE("%d", CAD_value[loraMode] * w);
      PRINTLN;

      delay(CAD_value[loraMode] * w);

      // check for free channel (SIFS/DIFS) once again
      startDoCad = millis();
      e = sx1272.doCAD(send_cad_number);
      endDoCad = millis();

      PRINT_CSTSTR("%s", "--> CAD duration ");
      PRINT_VALUE("%ld", endDoCad - startDoCad);
      PRINTLN;

      if (!e)
        PRINT_CSTSTR("%s", "OK2");
      else
        PRINT_CSTSTR("%s", "###2");

      PRINTLN;
    }
  }
  else
  {
    PRINT_CSTSTR("%s", "###1\n");

    // if we have "only once" behavior then exit here to not have retries
    if (onlyOnce)
      return 1;

      // wait for random number of DIFS
      #ifdef ARDUINO
        uint8_t w = random(1, 8);
      #else
        uint8_t w = rand() % 8 + 1;
      #endif

      PRINT_CSTSTR("%s", "--> waiting for ");
      PRINT_VALUE("%d", w);
      PRINT_CSTSTR("%s", " DIFS (DIFS=3SIFS) = ");
      PRINT_VALUE("%d", SIFS_value[loraMode] * 3 * w);
      PRINTLN;

      delay(SIFS_value[loraMode] * 3 * w);

      PRINT_CSTSTR("%s", "--> retry\n");
```

```
    } while (e);

    // CAD is OK, but need to check RSSI
    if (RSSIonSend)
    {

      e = sx1272.getRSSI();

      uint8_t rssi_retry_count = 10;

      if (!e)
      {

        PRINT_CSTSTR("%s", "--> RSSI ");
        PRINT_VALUE("%d", sx1272._RSSI);
        PRINTLN;

        while (sx1272._RSSI > -90 && rssi_retry_count)
        {

          delay(1);
          sx1272.getRSSI();
          PRINT_CSTSTR("%s", "--> RSSI ");
          PRINT_VALUE("%d", sx1272._RSSI);
          PRINTLN;
          rssi_retry_count--;
        }
      }
      else
        PRINT_CSTSTR("%s", "--> RSSI error\n");

      if (!rssi_retry_count)
        carrierSenseRetry = true;
      else
        carrierSenseRetry = false;
    }

  } while (carrierSenseRetry);
 }

 return 0;
}
```

ode yang digunakan pada penelitian ini dapat dilihat pada

:hub.com/DevyBadjarad/Program-Skripsi.git

Lampiran 2 *Source Code Gateway Multitasking*

```
// Program Name : Implementation of Multitasking in Edge Computing for Hydroponic Plant Monitoring Farm
// Contributor in This Program:
// 1. Conceptualize, Idea: Adnan, S.T., M.T., Ph.D.
// 2. Desain: Adnan, S.T., M.T., Ph.D. and Devy Noviani Badjarad
// 3. Implementation: Devy Noviani Badjarad
// 4. Testing: Devy Noviani Badjarad
// 5. Supervision: Adnan, S.T., M.T., Ph.D. and Dr. Eng. Zulkifli Tahir, S.T., M.Sc.

#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <WiFiMulti.h>
#include "SX1272.h"
#include "ThingSpeak.h"
#include <CircularBuffer.h>
#include <ESP32Time.h>
#include <string.h>

#define TS_ENABLE_SSL

#ifdef ARDUINO
// IMPORTANT when using an Arduino only. For a Raspberry-based gateway the distribution uses a radio.makefile file
/////////////////////////////////////////////////////////////////////////////
/////////////////////////////
// please uncomment only 1 choice
//
// uncomment if your radio is an HopeRF RFM92W, HopeRF RFM95W, Modtronix inAir9B, NiceRF1276
// or you known from the circuit diagram that output use the PABOOST line instead of the RFO line
#define PABOOST
/////////////////////////////////////////////////////////////////////////////
#endif
// IMPORTANT
/////////////////////////////////////////////////////////////////////////////
// please uncomment only 1 choice
//#define BAND868
#define BAND900
//#define BAND433
```

```
////////////////////////////////////////////////////////////////////////////
//////////////////////////
// For a Raspberry-based gateway the distribution uses a radio.makefile file that
can define MAX_DBM
//
#ifndef MAX_DBM
#define MAX_DBM 14
#endif

#ifndef LORA_PREAMBLE_LENGTH
#define LORA_PREAMBLE_LENGTH 8
#endif

#ifndef ARDUINO
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#ifdef GETOPT_ISSUE
int getopt(int argc, char *const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
#endif
#include <getopt.h>
#include <termios.h>
#include <signal.h>
#include <sys/time.h>
#include <time.h>
#include <math.h>
#include <string.h>
#endif

#ifdef ARDUINO
// and SPI library on Arduino platforms
#include <SPI.h>

#define PRINTLN Serial.println("")
#define PRINT_CSTSTR(fmt, param) Serial.print(F(param))
#define PRINT_STR(fmt, param) Serial.print(param)
#define PRINT_VALUE(fmt, param) Serial.print(param)
#define PRINT_HEX(fmt, param) Serial.print(param, HEX)
#define FLUSHOUTPUT Serial.flush();
#else
#define PRINTLN printf("\n")
#define PRINT_CSTSTR(fmt, param) printf(fmt, param)
 RINT_STR(fmt, param) PRINT_CSTSTR(fmt, param)
 RINT_VALUE(fmt, param) PRINT_CSTSTR(fmt, param)
 RINT_HEX(fmt, param) PRINT_VALUE(fmt, param)
 LUSHOUTPUT fflush(stdout);
```

```
#ifdef DEBUG
#define DEBUGLN PRINTLN
#define DEBUG_CSTSTR(fmt, param) PRINT_CSTSTR(fmt, param)
#define DEBUG_STR(fmt, param) PRINT_CSTSTR(fmt, param)
#define DEBUG_VALUE(fmt, param) PRINT_VALUE(fmt, param)
#else
#define DEBUGLN
#define DEBUG_CSTSTR(fmt, param)
#define DEBUG_STR(fmt, param)
#define DEBUG_VALUE(fmt, param)
#endif

int count1;
int count2;
int count3;

void task1(void *pvParameters);
void task2(void *pvParameters);
void task3(void *pvParameters);

void setup() {
  int e;
  #ifdef ARDUINO
  delay(3000);
  randomSeed(analogRead(14));

  // Open serial communications and wait for port to open:
  #ifdef __SAMD21G18A__
  SerialUSB.begin(38400);
  #else
  Serial.begin(38400);
  #endif

  //CONNECTED TO WIFI
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  while (wifiMulti.run() != WL_CONNECTED)
  {
    delay(1000);
  }
  Serial.println("Connected");
  Serial.println(WiFi.localIP());

        eak.begin(client); // Initialize ThingSpeak



        eate(
```

```
    task1,
    "task1",
    10000,
    NULL,
    1,
    NULL
  );

  xTaskCreate(
    task2,
    "task2",
    10000,
    NULL,
    1,
    NULL
  );

  xTaskCreate(
    task3,
    "task3",
    10000,
    NULL,
    1,
    NULL
  );

  vTaskStartScheduler();

  #if defined ARDUINO && defined SHOW_FREEMEMORY && not defined __MK20DX256__ && not
defined   __MKL26Z64__   &&   not   defined   __SAMD21G18A__   &&   not   defined
_VARIANT_ARDUINO_DUE_X_
    // Print a start message
    Serial.print(freeMemory());
    Serial.println(F(" bytes of free memory."));
  #endif

  #else
    srand(time(NULL));
  #endif

  // Power ON the module
  e = sx1272.ON();

  PRINT_CSTSTR("%s", "^$**********Power ON: state ");
      ALUE("%d", e);
      ;
```

```
    radioON = true;
    startConfig();
  }

  e = sx1272.getSyncWord();

  if (!e)
  {
    PRINT_CSTSTR("%s", "^$Default sync word: 0x");
    PRINT_HEX("%X", sx1272._syncWord);
    PRINTLN;
  }

  if (optSW != 0x12)
  {
    e = sx1272.setSyncWord(optSW);

    PRINT_CSTSTR("%s", "^$Set sync word to 0x");
    PRINT_HEX("%X", optSW);
    PRINTLN;
    PRINT_CSTSTR("%s", "^$LoRa sync word: state ");
    PRINT_VALUE("%d", e);
    PRINTLN;
  }

  FLUSHOUTPUT;
  delay(1000);

  #ifdef LORA_LAS
  loraLAS.ON(LAS_ON_WRESET);
  #endif

  #ifdef CAD_TEST
  PRINT_CSTSTR("%s", "Do CAD test\n");
  #endif

  #if not defined ARDUINO && defined DOWNLINK

  lastDownlinkCheckTime = millis();

  #endif

}
```

ld use the CarrierSense function added in the SX1272 library, but it is
enient to duplicate it here
t we could easily modify it for testing

```
// in v1.5 the "only once" behavior is implemented for the gateway when it transmit
downlink packets
// to avoid blocking the gateway on a busy channel. Therefore from v1.5 the
implementation differs from the
// carrier sense function added in the SX1272 library
//
int CarrierSense(bool onlyOnce = false) {

  int e;
  bool carrierSenseRetry = false;

  if (send_cad_number)
  {
    do
    {
      do
      {
        // check for free channel (SIFS/DIFS)
        startDoCad = millis();
        e = sx1272.doCAD(send_cad_number);
        endDoCad = millis();

        PRINT_CSTSTR("%s", "--> CAD duration ");
        PRINT_VALUE("%ld", endDoCad - startDoCad);
        PRINTLN;

        if (!e)
        {
          PRINT_CSTSTR("%s", "OK1\n");

          if (extendedIFS)
          {
            // wait for random number of CAD
            #ifdef ARDUINO
              uint8_t w = random(1, 8);
            #else
              uint8_t w = rand() % 8 + 1;
            #endif

            PRINT_CSTSTR("%s", "--> waiting for ");
            PRINT_VALUE("%d", w);
            PRINT_CSTSTR("%s", " CAD = ");
            PRINT_VALUE("%d", CAD_value[loraMode] * w);
            PRINTLN;

            delay(CAD_value[loraMode] * w);

            // check for free channel (SIFS/DIFS) once again
            startDoCad = millis();
```

```
      e = sx1272.doCAD(send_cad_number);
      endDoCad = millis();

      PRINT_CSTSTR("%s", "--> CAD duration ");
      PRINT_VALUE("%ld", endDoCad - startDoCad);
      PRINTLN;

      if (!e)
        PRINT_CSTSTR("%s", "OK2");
      else
        PRINT_CSTSTR("%s", "###2");

      PRINTLN;
    }
  }
  else
  {
    PRINT_CSTSTR("%s", "###1\n");

    // if we have "only once" behavior then exit here to not have retries
    if (onlyOnce)
      return 1;

      // wait for random number of DIFS
      #ifdef ARDUINO
        uint8_t w = random(1, 8);
      #else
        uint8_t w = rand() % 8 + 1;
      #endif

      PRINT_CSTSTR("%s", "--> waiting for ");
      PRINT_VALUE("%d", w);
      PRINT_CSTSTR("%s", " DIFS (DIFS=3SIFS) = ");
      PRINT_VALUE("%d", SIFS_value[loraMode] * 3 * w);
      PRINTLN;

      delay(SIFS_value[loraMode] * 3 * w);

      PRINT_CSTSTR("%s", "--> retry\n");
  }

} while (e);

// CAD is OK, but need to check RSSI
   (RSSIonSend)


    = sx1272.getRSSI();
```

```
        uint8_t rssi_retry_count = 10;

        if (!e)
        {
          PRINT_CSTSTR("%s", "--> RSSI ");
          PRINT_VALUE("%d", sx1272._RSSI);
          PRINTLN;

          while (sx1272._RSSI > -90 && rssi_retry_count)
          {
            delay(1);
            sx1272.getRSSI();
            PRINT_CSTSTR("%s", "--> RSSI ");
            PRINT_VALUE("%d", sx1272._RSSI);
            PRINTLN;
            rssi_retry_count--;
          }
        }
        else
          PRINT_CSTSTR("%s", "--> RSSI error\n");

        if (!rssi_retry_count)
          carrierSenseRetry = true;
        else
          carrierSenseRetry = false;
      }
    } while (carrierSenseRetry);
  }

  return 0;
}
```

*Source cod*e yang digunakan pada penelitian ini dapat dilihat pada https://github.com/DevyBadjarad/Program-Skripsi.git

Lampiran 3 *Source Code Node* Sensor 3

```
// IMPORTANT
////////////////////////////////////////////////////////////////////////////
//////////////////////////////
// add here some specific board define statements if you want to implement user-
defined specific settings
// A/ LoRa radio node from IoTMCU: https://www.tindie.com/products/IOTMCU/lora-
radio-node-v10/
//#define IOTMCU_LORA_RADIO_NODE
////////////////////////////////////////////////////////////////////////////
//////////////////////////////

#include <SPI.h>
// Include the SX1272
#include "SX1272.h"
// Include sensors
#include "Sensor.h"
#include "DHT11_Humidity.h"
#include "DHT11_Temperature.h"
#include "DHT.H"
//#include "DS18B20.h"
#include "HCSR04.h"


// IMPORTANT
////////////////////////////////////////////////////////////////////////////
//////////////////////////////
// please uncomment only 1 choice
//
#define ETSI_EUROPE_REGULATION
//#define FCC_US_REGULATION
//#define SENEGAL_REGULATION
////////////////////////////////////////////////////////////////////////////

// IMPORTANT
////////////////////////////////////////////////////////////////////////////
// please uncomment only 1 choice
//#define BAND868
#define BAND900
//#define BAND433
////////////////////////////////////////////////////////////////////////////

#ifdef ETSI_EUROPE_REGULATION
    MAX_DBM 14
    ous way for setting output power
    powerLevel='M';
    fined SENEGAL REGULATION
```

```
#define MAX_DBM 10
// previous way for setting output power
// 'H' is actually 6dBm, so better to use the new way to set output power
// char powerLevel='H';
#endif

#ifdef BAND868
#ifdef SENEGAL_REGULATION
const uint32_t DEFAULT_CHANNEL=CH_04_868;
#else
const uint32_t DEFAULT_CHANNEL=CH_10_868;
#endif
#elif defined BAND900
const uint32_t DEFAULT_CHANNEL=CH_12_900;
#elif defined BAND433
const uint32_t DEFAULT_CHANNEL=CH_00_433;
#endif

void setup()
{
  int e;

#ifdef LOW_POWER
  bool low_power_status = IS_LOWPOWER;
#ifdef __SAMD21G18A__
  rtc.begin();
#endif
#else
  bool low_power_status = IS_NOT_LOWPOWER;
#endif

////////////////////////////////////////////////////////////////////
// ADD YOUR SENSORS HERE
// Sensor(nomenclature, is_analog, is_connected, is_low_power, pin_read, pin_power,
pin_trigger=-1)
  sensor_ptrs[0] = new DHT11_Temperature((char*)"TC1", IS_NOT_ANALOG, IS_CONNECTED,
low_power_status, (uint8_t) 2, (uint8_t) 7 /*no pin trigger*/);
  sensor_ptrs[1] = new DHT11_Humidity((char*)"HU1", IS_NOT_ANALOG, IS_CONNECTED,
low_power_status, (uint8_t) 2, (uint8_t) 7 /*no pin trigger*/);
  sensor_ptrs[2] = new HCSR04((char*)"DIS", IS_NOT_ANALOG, IS_CONNECTED,
low_power_status, (uint8_t) 3, (uint8_t) 6, (uint8_t) 8); // for the MEGA
////////////////////////////////////////////////////////////////////

  delay(1000);
       serial communications and wait for port to open:
    ed __SAMD21G18A__ && not defined ARDUINO_SAMD_FEATHER_M0
    3B.begin(38400);

    begin(38400);
```

```
#endif
  // Print a start message
  PRINT_CSTSTR("%s","Generic LoRa sensor\n");

#ifdef ARDUINO_AVR_PRO
  PRINT_CSTSTR("%s","Arduino Pro Mini detected\n");
#endif
#ifdef ARDUINO_AVR_NANO
  PRINT_CSTSTR("%s","Arduino Nano detected\n");
#endif
#ifdef ARDUINO_AVR_MINI
  PRINT_CSTSTR("%s","Arduino MINI/Nexus detected\n");
#endif
#ifdef ARDUINO_AVR_MEGA2560
  PRINT_CSTSTR("%s","Arduino Mega2560 detected\n");
#endif
#ifdef ARDUINO_SAM_DUE
  PRINT_CSTSTR("%s","Arduino Due detected\n");
#endif
#ifdef __MK66FX1M0__
  PRINT_CSTSTR("%s","Teensy36 MK66FX1M0 detected\n");
#endif
#ifdef __MK64FX512__
  PRINT_CSTSTR("%s","Teensy35 MK64FX512 detected\n");
#endif
#ifdef __MK20DX256__
  PRINT_CSTSTR("%s","Teensy31/32 MK20DX256 detected\n");
#endif
#ifdef __MKL26Z64__
  PRINT_CSTSTR("%s","TeensyLC MKL26Z64 detected\n");
#endif
#if defined ARDUINO_SAMD_ZERO && not defined ARDUINO_SAMD_FEATHER_M0
  PRINT_CSTSTR("%s","Arduino M0/Zero detected\n");
#endif
#ifdef ARDUINO_AVR_FEATHER32U4
  PRINT_CSTSTR("%s","Adafruit Feather32U4 detected\n");
#endif
#ifdef  ARDUINO_SAMD_FEATHER_M0
  PRINT_CSTSTR("%s","Adafruit FeatherM0 detected\n");
#endif

// See http://www.nongnu.org/avr-libc/user-manual/using_tools.html
// for the list of define from the AVR compiler

AVR_ATmega328P__
3TSTR("%s","ATmega328P detected\n");

AVR_ATmega32U4__
3TSTR("%s","ATmega32U4 detected\n");
```

```
#endif
#ifdef __AVR_ATmega2560__
  PRINT_CSTSTR("%s","ATmega2560 detected\n");
#endif
#ifdef __SAMD21G18A__
  PRINT_CSTSTR("%s","SAMD21G18A ARM Cortex-M0+ detected\n");
#endif
#ifdef __SAM3X8E__
  PRINT_CSTSTR("%s","SAM3X8E ARM Cortex-M3 detected\n");
#endif

  // Power ON the module
  sx1272.ON();

#ifdef WITH_EEPROM
  // get config from EEPROM
  EEPROM.get(0, my_sx1272config);

  // found a valid config?
  if (my_sx1272config.flag1==0x12 && my_sx1272config.flag2==0x35) {
    PRINT_CSTSTR("%s","Get back previous sx1272 config\n");

    // set sequence number for SX1272 library
    sx1272._packetNumber=my_sx1272config.seq;
    PRINT_CSTSTR("%s","Using packet sequence number of ");
    PRINT_VALUE("%d", sx1272._packetNumber);
    PRINTLN;

#ifdef FORCE_DEFAULT_VALUE
    PRINT_CSTSTR("%s","Forced to use default parameters\n");
    my_sx1272config.flag1=0x12;
    my_sx1272config.flag2=0x35;
    my_sx1272config.seq=sx1272._packetNumber;
    my_sx1272config.addr=node_addr;
    my_sx1272config.idle_period=idlePeriodInMin;
    my_sx1272config.overwrite=0;
    EEPROM.put(0, my_sx1272config);
#else
    // get back the node_addr
    if (my_sx1272config.addr!=0 && my_sx1272config.overwrite==1) {

        PRINT_CSTSTR("%s","Used stored address\n");
        node_addr=my_sx1272config.addr;
    }

     RINT_CSTSTR("%s","Stored node addr is null\n");

     t back the idle period
     y_sx1272config.idle_period!=0 && my_sx1272config.overwrite==1) {
```

```
        PRINT_CSTSTR("%s","Used stored idle period\n");
        idlePeriodInMin=my_sx1272config.idle_period;
    }
    else
        PRINT_CSTSTR("%s","Stored idle period is null\n");
#endif

    PRINT_CSTSTR("%s","Using node addr of ");
    PRINT_VALUE("%d", node_addr);
    PRINTLN;

    PRINT_STR("%s","Using idle period of ");
    PRINT_VALUE("%d", idlePeriodInMin);
    PRINTLN;
  }
  else {
    // otherwise, write config and start over
    my_sx1272config.flag1=0x12;
    my_sx1272config.flag2=0x35;
    my_sx1272config.seq=sx1272._packetNumber;
    my_sx1272config.addr=node_addr;
    my_sx1272config.idle_period=idlePeriodInMin;
    my_sx1272config.overwrite=0;
  }
#endif

  // Set transmission mode and print the result
  e = sx1272.setMode(loraMode);
  PRINT_CSTSTR("%s","Setting Mode: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;

#ifdef MY_FREQUENCY
  e = sx1272.setChannel(MY_FREQUENCY*1000000.0*RH_LORA_FCONVERT);
  PRINT_CSTSTR("%s","Setting customized frequency: ");
  PRINT_VALUE("%f", MY_FREQUENCY);
  PRINTLN;
#else
  e = sx1272.setChannel(DEFAULT_CHANNEL);
#endif
  PRINT_CSTSTR("%s","Setting Channel: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;
```

```
    le carrier sense
    _enableCarrierSense=true;
    N_POWER
    : with low power, when setting the radio module in sleep mode
```

```
  // there seem to be some issue with RSSI reading
  sx1272._RSSIonSend=false;
#endif

#ifdef USE_LORAWAN_SW
  e = sx1272.setSyncWord(0x34);
  PRINT_CSTSTR("%s","Set sync word to 0x34: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;
#endif

  // Select amplifier line; PABOOST or RFO
#ifdef PABOOST
  sx1272._needPABOOST=true;
#endif

  e = sx1272.setPowerDBM((uint8_t)MAX_DBM);
  PRINT_CSTSTR("%s","Setting Power: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;

  // Set the node address and print the result
  e = sx1272.setNodeAddress(node_addr);
  PRINT_CSTSTR("%s","Setting node addr: state ");
  PRINT_VALUE("%d", e);
  PRINTLN;

  // Print a success message
  PRINT_STR("%s","SX1272 successfully configured\n");

  delay(500);
}
```

*Source cod*e yang digunakan pada penelitian ini dapat dilihat pada https://github.com/DevyBadjarad/Program-Skripsi.git

Lampiran 4 *Source Code Node* Sensor 2

```
// Include the SX1272
#include "SX1272.h"
#include "my_demo_sensor_code.h"

// IMPORTANT SETTINGS
///////////////////////////////////////////////////////////////////////////
////////////////////////////
//
// uncomment if your radio is an HopeRF RFM92W, HopeRF RFM95W, Modtronix
inAir9B, NiceRF1276
// or you known from the circuit diagram that output use the PABOOST line
instead of the RFO line

#define PABOOST
///////////////////////////////////////////////////////////////////////////
#include <Wire.h>
///////////////////////////////////////////////////////////////////
// CHANGE HERE THE NODE ADDRESS

#define node_addr 6
#define pH_sensor_pin A2
///////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////
// UNCOMMENT HERE IF YOU HAVE CONNECTED A SMALL OLED SCREEN

//#define OLED
///////////////////////////////////////////////////////////////////

// IMPORTANT
///////////////////////////////////////////////////////////////////////////
// please uncomment only 1 choice
//
#define ETSI_EUROPE_REGULATION
//#define FCC_US_REGULATION
//#define SENEGAL_REGULATION
///////////////////////////////////////////////////////////////////////////

#ifdef ETSI_EUROPE_REGULATION
#define MAX_DBM 14
// previous way for setting output power
// char powerLevel='M';
#elif defined SENEGAL_REGULATION
#define MAX_DBM 10
// previous way for setting output power
// 'H' is actually 6dBm, so better to use the new way to set output power
// char powerLevel='H';
#elif defined FCC_US_REGULATION
#define MAX_DBM 14
#endif

#ifdef OLED
//you can also power the OLED screen with a digital pin, here pin 8
#define OLED_PWR_PIN 8
#include <U8x8lib.h>
ction may depend on the board. Use A5/A4 for most Arduino boards. On
based board we use GPI05 and GPI04. Heltec ESP32 has embedded OLED.
ned ARDUINO_Heltec_WIFI_LoRa_32 || defined ARDUINO_WIFI_LoRa_32 ||
HELTEC_LORA
```

```
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 15, /* data=*/ 4, /* reset=*/
16);
#elif defined ESP8266 || defined ARDUINO_ESP8266_ESP01
//U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 5, /* data=*/ 4, /* reset=*/
U8X8_PIN_NONE);
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 12, /* data=*/ 14, /* reset=*/
U8X8_PIN_NONE);
#else
//reset is not used
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ A5, /* data=*/ A4, /* reset=*/
U8X8_PIN_NONE);
#endif
char oled_msg[20];
#endif

#define FLOAT_TEMP

#define DEFAULT_DEST_ADDR 1
#define LORAMODE  1

double sensor_value;
unsigned long nextTransmissionTime=0L;
char float_str[20];
uint8_t message[100];

int loraMode=LORAMODE;

char *ftoa(char *a, double f, int precision)
{
 long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

 char *ret = a;
 long heiltal = (long)f;
 itoa(heiltal, a, 10);
 while (*a != '\0') a++;
 *a++ = '.';
 long desimal = abs((long)((f - heiltal) * p[precision]));
 if (desimal < p[precision-1]) {
  *a++ = '0';
 }
 itoa(desimal, a, 10);
 return ret;
}


void setup()
{
  int e;
  Wire.begin();

  delay(3000);
  // Open serial communications and wait for port to open:
  Serial.begin(38400);
  // Print a start message
  Serial.println(F("Simple LoRa sensor demo"));

   pinMode(pH_sensor_pin, INPUT);

   ED
   gin();
   tFont(u8x8_font_chroma48medium8_r);
   awString(0, 1, "Simple TempDemo");
   awString(0, 2, "LoRa mode 1");
```

```
#if defined ARDUINO_Heltec_WIFI_LoRa_32 || defined ARDUINO_WIFI_LoRa_32 || defined
HELTEC_LORA
  // for the Heltec ESP32 WiFi LoRa module
  sx1272.setCSPin(18);
#endif

  // Power ON the module
  sx1272.ON();

  // Set transmission mode and print the result
  e = sx1272.setMode(loraMode);
  Serial.print(F("Setting Mode: state "));
  Serial.println(e, DEC);

  // enable carrier sense
  sx1272._enableCarrierSense=true;

  // Select frequency channel
  e = sx1272.setChannel(CH_12_900);
  Serial.print(F("Setting Channel: state "));
  Serial.println(e, DEC);

  // Select amplifier line; PABOOST or RFO
#ifdef PABOOST
  sx1272._needPABOOST=true;
#endif

  e = sx1272.setPowerDBM((uint8_t)MAX_DBM);

  Serial.print(F("Setting Power: state "));
  Serial.println(e);

  // Set the node address and print the result
  e = sx1272.setNodeAddress(node_addr);
  Serial.print(F("Setting node addr: state "));
  Serial.println(e, DEC);

  // Print a success message
  Serial.println(F("SX1272 successfully configured"));

  delay(500);

  // initialization of the demo sensor
  sensor_Init();
}

          float simulate_pH_reading() {
  // Menghasilkan nilai pH acak antara 6.0 hingga 7.0
  return 6.0 + (random(0, 101) / 100.0); // Menghasilkan nilai antara 6.0 hingga
7.0
}
```

*Source cod*e yang digunakan pada penelitian ini dapat dilihat pada https://github.com/DevyBadjarad/Program-Skripsi.git

Lampiran 5 *Source Code Node* Sensor 3

```
// Include the SX1272
#include "SX1272.h"
#include "my_demo_sensor_code.h"

// IMPORTANT SETTINGS
///////////////////////////////////////////////////////////////////////////
//
// uncomment if your radio is an HopeRF RFM92W, HopeRF RFM95W, Modtronix
inAir9B, NiceRF1276
// or you known from the circuit diagram that output use the PABOOST line
instead of the RFO line

#define PABOOST
///////////////////////////////////////////////////////////////////////////
#include <Wire.h>
///////////////////////////////////////////////////////////////////
// CHANGE HERE THE NODE ADDRESS

#define node_addr 5
///////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////
// UNCOMMENT HERE IF YOU HAVE CONNECTED A SMALL OLED SCREEN

//#define OLED
///////////////////////////////////////////////////////////////////

// IMPORTANT
///////////////////////////////////////////////////////////////////////////
// please uncomment only 1 choice
//
#define ETSI_EUROPE_REGULATION
//#define FCC_US_REGULATION
//#define SENEGAL_REGULATION
///////////////////////////////////////////////////////////////////////////

#ifdef ETSI_EUROPE_REGULATION
#define MAX_DBM 14
// previous way for setting output power
// char powerLevel='M';
#elif defined SENEGAL_REGULATION
#define MAX_DBM 10
// previous way for setting output power
// 'H' is actually 6dBm, so better to use the new way to set output power
// char powerLevel='H';
#elif defined FCC_US_REGULATION
#define MAX_DBM 14
#endif

#ifdef OLED
//you can also power the OLED screen with a digital pin, here pin 8
#define OLED_PWR_PIN 8
#include <U8x8lib.h>
// connection may depend on the board. Use A5/A4 for most Arduino boards. On
ESP8266-based board we use GPI05 and GPI04. Heltec ESP32 has embedded OLED.
      ned ARDUINO_Heltec_WIFI_LoRa_32 || defined ARDUINO_WIFI_LoRa_32 ||
      HELTEC_LORA
      1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 15, /* data=*/ 4, /*
       16);
      fined ESP8266 || defined ARDUINO_ESP8266_ESP01
```

```
//U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 5, /* data=*/ 4, /* reset=*/
U8X8_PIN_NONE);
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ 12, /* data=*/ 14, /* reset=*/
U8X8_PIN_NONE);
#else
//reset is not used
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ A5, /* data=*/ A4, /* reset=*/
U8X8_PIN_NONE);
#endif
char oled_msg[20];
#endif

#define FLOAT_TEMP

#define DEFAULT_DEST_ADDR 1
#define LORAMODE  1

double sensor_value;
unsigned long nextTransmissionTime=0L;
char float_str[20];
uint8_t message[100];

int loraMode=LORAMODE;

char *ftoa(char *a, double f, int precision)
{
 long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

 char *ret = a;
 long heiltal = (long)f;
 itoa(heiltal, a, 10);
 while (*a != '\0') a++;
 *a++ = '.';
 long desimal = abs((long)((f - heiltal) * p[precision]));
 if (desimal < p[precision-1]) {
  *a++ = '0';
 }
 itoa(desimal, a, 10);
 return ret;
}

void setup()
{
  int e;
  Wire.begin();

  delay(3000);
  // Open serial communications and wait for port to open:
  Serial.begin(38400);
  // Print a start message
  Serial.println(F("Simple LoRa sensor demo"));

#ifdef OLED
  u8x8.begin();
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  u8x8.drawString(0, 1, "Simple TempDemo");
  u8x8.drawString(0, 2, "LoRa mode 1");
#endif

ed ARDUINO_Heltec_WIFI_LoRa_32 || defined ARDUINO_WIFI_LoRa_32 || defined
RA
the Heltec ESP32 WiFi LoRa module
setCSPin(18);
```

```
  // Power ON the module
  sx1272.ON();

  // Set transmission mode and print the result
  e = sx1272.setMode(loraMode);
  Serial.print(F("Setting Mode: state "));
  Serial.println(e, DEC);

  // enable carrier sense
  sx1272._enableCarrierSense=true;

  // Select frequency channel
  e = sx1272.setChannel(CH_12_900);
  Serial.print(F("Setting Channel: state "));
  Serial.println(e, DEC);

  // Select amplifier line; PABOOST or RFO
#ifdef PABOOST
  sx1272._needPABOOST=true;
#endif

  e = sx1272.setPowerDBM((uint8_t)MAX_DBM);

  Serial.print(F("Setting Power: state "));
  Serial.println(e);

  // Set the node address and print the result
  e = sx1272.setNodeAddress(node_addr);
  Serial.print(F("Setting node addr: state "));
  Serial.println(e, DEC);

  // Print a success message
  Serial.println(F("SX1272 successfully configured"));

  delay(500);

  // initialization of the demo sensor
  sensor_Init();
}

float simulate_TDS_reading() {
  // Menghasilkan nilai TDS simulasi antara 1200 hingga 1230
  return random(1200, 1231); // Menghasilkan nilai antara 1200 hingga 1230
}
```

*Source cod*e yang digunakan pada penelitian ini dapat dilihat pada https://github.com/DevyBadjarad/Program-Skripsi.git

Lampiran 6 Lembar Perbaikan Skripsi

LEMBAR PERBAIKAN SKRIPSI

"PENERAPAN MULTITASKING PADA EDGE COMPUTING UNTUK MONITORING TANAMAN HIDROPONIK FARM"

OLEH:

DEVY NOVIANI BADJARAD
D121171014

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 13 Desember 2023.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Adnan, S.T., M.T., Ph.D.	
Sekretaris	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.	
Anggota	Dr. Eng. Muhammad Niswar, S.T., M.IT.	
	Dr. Eng. Ady Wahyudi Paundu, S.T., M.T.	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Adnan, S.T., M.T., Ph.D.	
II	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.	