# BAB V
# PENUTUP

## 5.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan telah berhasil dibuat alat Pegaman Pintu Menggunakan Kode Dan Modul GSM Berbasis Arduino. Alat pengaman ini bekerja sesuai yang diharapkan antara lain sebagai berikut.

- Alat dapat mengirim pesan kepada *ADMIN* ketika terjadi pembobolan pembukaan pintu.
- Akses *USER* hanya bisa membuka kunci pintu.
- Akses *ADMIN* diprogram dapat mengatur no tujuan pesan, menganti kode *USER*, dan menganti kode *ADMIN*.
- Alat dilengkapi dengan sensor magnet untuk mendeteksi pembobolan pembukaan kunci pintu.
- Alarm akan berbunyi ketika terjadi kesalahan input kode akses dan pada saat terjadi pembobolan pembukaan kunci pintu.

## 5.2 Saran

Pada hasil penelitian yang telah dilakukan terdapat beberapa saran yang kiranya dapat bermanfaat antara lain.

1. Sebaiknya ditambakan cadangan pawer suplay pada saat listrik padam.
2. Sebaiknya mode akses admin dapat diprogram bisa akses melalui SMS
3. Penberitahuan informasi pulsa atau masa aktif kartu pada alat sebaiknya di program bisa dikirim langsung ke pada admin.

## Daftar Pustaka


Adreson. J.2007. Pengendalian Pintu Gerbang Menggunakan Mikrokontroler AT89S8252. Skripsi.Medan : Defartemen Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sumatra Barat.

Agusta.2012. Sistim Proteksi Brangkas Berpassword Menggunakan Magnetik Door lock Sebagai Pengerak Doorstrike Berbasis Mikrokontroler.Tugas Ahir.Untuk Memperoleh Gelar Ahli Madya.Program Diploma III Teknik Elektro Jurusan Teknik Elektro.Fakultas Teknik .Universitas Negeri Semarang.

Bana,2015.Aplikasi Android Untuk Mengatur Switch Pada Perangkat Elektronik Nirkabel.Makalah.Program Studi Informatika Fakultas Komunikasi Dan Informatika. Universitas Muhammadiyah Surakarta.

Ginanjar.2012.Perancangan dan Implementasi Otomatisasi Kunci Mobil dengan Kontrol Bluetooth Menggunakan Mobile Aplication Berbasis Adroid. Skripsi. Teknik Telekomunikasi Fakultas Ilmu Terapan Universitas Telkom.

Helmi. 2013. Rancang Bangun Magnetik Door Lock Menggunakan Keypad Dan Selenoid Berbasis Mikrokontroler Arduino Uno. Skripsi.Program Studi Teknik Elektro. Bandung: FPTK Universitas Pendidikan Indonesia.

Karseno Doni,2011. Security Password Menggunakan Remote Berbasis Mikrokontroler Arduino. Jurusan Teknik Informatika STIMIK AMIKOM YOGYAKARTA

Ramakumbo Gusti Ario,2011. Magnetik Door Lock Menggunakan Kode Pengaman Berbasis Atmega 328. Skripsi.yogyakarta : Jurusan Pendidikan Teknik Fakultas Teknik Universitas Negeri Yogyakarta.

Situmorang D. Erikson,2012. Rancang Bangun Alat Buka Tutup Pintu Pagar Dengan Menggunakan Handphone Dan Keypad. Teknik Elektro STMIK FT UNSRAT.

Wicaksono Putut,2014. Sistim Aplikasi Kunci Dengan Kode Password Berbasis Mikrokontroler AT Mega 16. Skripsi. Program Studi Teknik Elektro Fakultas Teknik. Universitas Muhammadiyah Ponorogo.

https://elektroku.com/rangkaian-driver-relay-sederhana. Di akses 12 Februari 2018.

https://github.com/ZulNs/MultitapKeypad. Di akses 12 Februari 2018

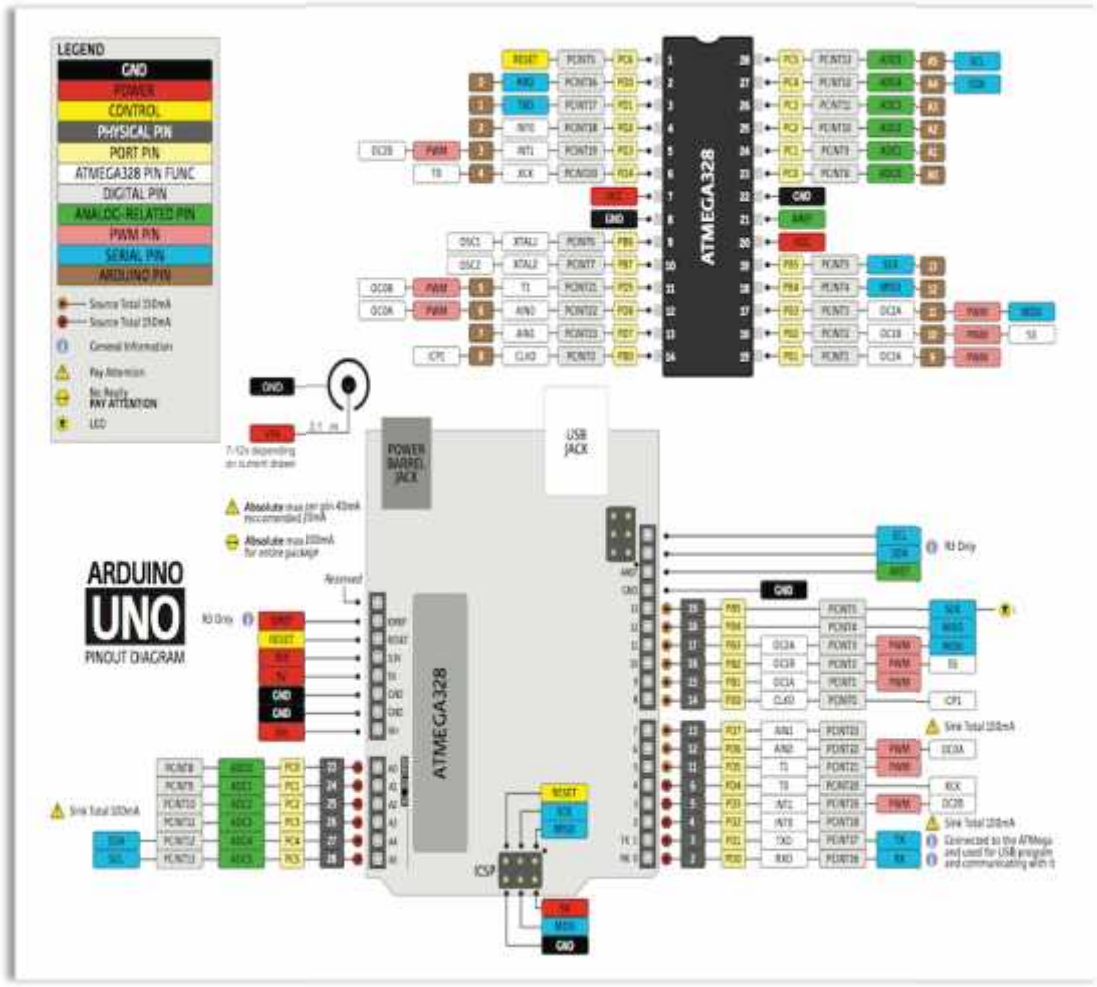
https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library. Di akses 12 Februari 2018.

https://www.pjrc.com/teensy/td_libs_SoftwareSerial.html.Di akses 17 Maret 2018.

https://purnomosejati.wordpress.com/2011/08/25/mengenal-komunikasi-i2cinter-integrated-circuit/. Di akses 17 Maret 2018.

# LAMPIRAN

**Lampiran 1** Peta Pin Arduino Uno



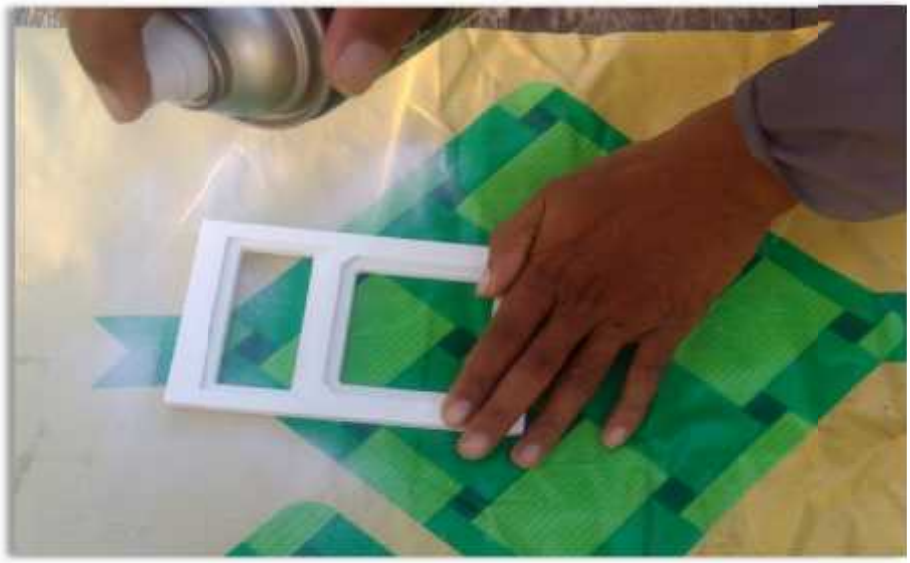
**Lampiran 2** Pemograman

Pemoggraman Keypad

Pemograman LCD

**Lampiran 3** Pembuatan Box Alat

Pembuatan Lubang Bout



Pengecatan Box Alat

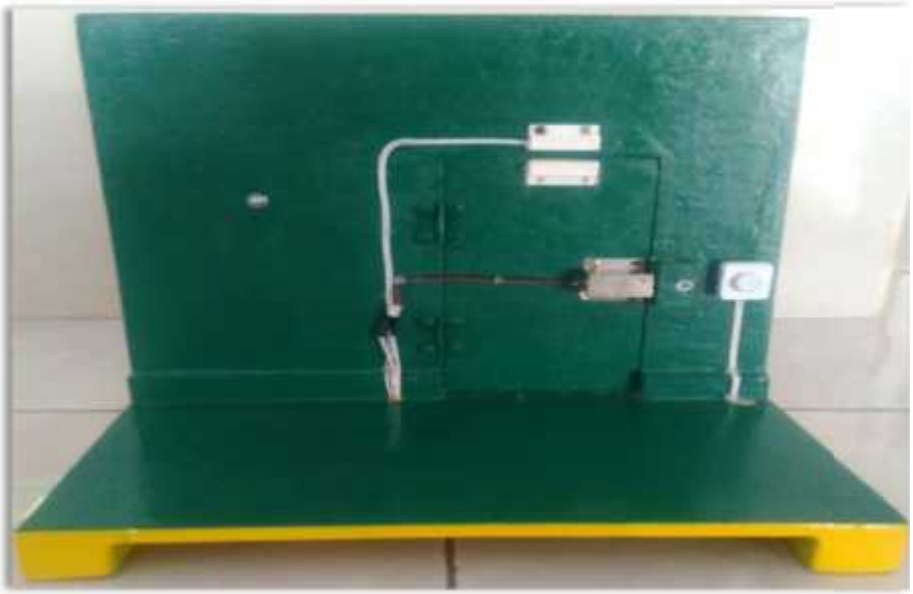
**Lampiran 4** Desain Miniatur Box Alat


Bagian Depan Box Alat

Bagian Dalam Box Alat

**Lampiran 5** Desain Miniatur Alat Pengaman

Tampak Depan Sistim Pengaman



Tampak Depan Sistim Pengaman

**Lampiran 6** Lisning Program

```
1.  //Pengaman Pintu Menggunakan Kode Dan Modul GSM Berbasis Arduino
2.  #include <EEPROM.h>
3.  #include <MultitapKeypad.h>
4.  #include <LiquidCrystal_I2C.h>
5.  #include <Sim800L.h>
6.  #include <gprs.h>
7.  #include <SoftwareSerial.h>
8.  GPRS gprs;
9.  SoftwareSerial SIM800L(7, 8); // RX,TX
10. #define DOOR_STATE_LOCKED       0
11. #define DOOR_STATE_UNLOCKED     1
12. #define DOOR_STATE_OPENED       2
13. #define UNLOCKED_PERIOD         20000
14. #define SLEEP_PERIOD            20000
15. #define DOOR_CLOSE_SWITCH_PIN   2     // Sense by INT0
16. #define DOOR_UNLOCK_SWITCH_PIN  3     // Sense by INT1
17. #define DOOR_LOCKER_PIN         4
18. #define BUZZER_PIN              5
19. #define ROW4  A0
20. #define ROW3  A1
21. #define ROW2  A2
22. #define ROW1  A3
23. #define COL4  9
24. #define COL3  10
25. #define COL2  11
26. #define COL1  12
27. #define CHR_BOUND       3
28. #define BACKSPACE       8
29. #define CLEARSCREEN     12
30. #define CARRIAGE_RETURN  13
31. #define CAPSLOCK_ON     17
32. #define CAPSLOCK_OFF    18
33. #define NUMLOCK_ON      19
34. #define NUMLOCK_OFF     20
35. // Multitap Symbols from '1' key
36. const char SYMBOL_1[] PROGMEM =
```

```
{
  '(', ')', '[', ']', '{',
  '}', '@', '$', CHR_BOUND
};
// Multitap Symbols from '*' key
const char SYMBOL_A[] PROGMEM =
{
  '/', '+', '-', '=', '%',
'^', '_', CHR_BOUND
};
// Multitap Symbols from '#' key
const char SYMBOL_NS[] PROGMEM =
{
',', '.',  ';', ':', '!',
  '?', '\'', '"', CHR_BOUND
};
const char ALPHABET[] PROGMEM =
{
'A', 'B', 'C', CHR_BOUND, CHR_BOUND,
'D', 'E', 'F', CHR_BOUND, CHR_BOUND,
'G', 'H', 'I', CHR_BOUND, CHR_BOUND,
  'J', 'K', 'L', CHR_BOUND, CHR_BOUND,
'M', 'N', 'O', CHR_BOUND, CHR_BOUND,
  'P', 'Q', 'R', 'S',       CHR_BOUND,
  'T', 'U', 'V', CHR_BOUND, CHR_BOUND,
'W', 'X', 'Y', 'Z',       CHR_BOUND
};
const int EEPROM_ADMIN_PASSWORD = EEPROM.length() - 48;
const int EEPROM_USER_PASSWORD  = EEPROM.length() - 32;
const int EEPROM_PHONE_NUMBER   = EEPROM.length() - 16;
// creates lcd as LiquidCrystal object
LiquidCrystal_I2C lcd(0x3f, 16, 2);
// creates kpd as MultitapKeypad object
// for matrix 4 x 3 keypad
// MultitapKeypad kpd(ROW0, ROW1, ROW2, ROW3, COL0, COL1, COL2);
// for matrix 4 x 4 keypad
MultitapKeypad kpd(ROW1, ROW2, ROW3, ROW4, COL1, COL2, COL3,
```

```
74. COL4);
75. // creates key as Key object
76. Key key;
77. boolean isAlphaMode = true;
78. boolean isUpperCaseMode = true;
79. boolean isEndOfDisplay = false;
80. boolean isWaitingForSleep = false;
81. byte startCursorPos;
82. byte endCursorPos;
83. byte cursorPos;
84. byte chrCtr;
85. char strAdminPassword[16] = "admin";
86. char strUserPassword[16] = "user";
87. char strPhoneNumber[16] = "085395262767";
88. char strBuffer[16];
89. volatile boolean isInt0 = false;
90. volatile boolean isInt1 = false;
91. byte doorState;
92. long unlockedTimeout;
93. long sleepTimeout;
94. void setup() {
95.  Serial.begin(9600);
96. while(!Serial);
97. Serial.println("SIM800L Demo Send SMS via Seeeduino");
98. gprs.preInit();
99. delay(1000);
100.     while(0 != gprs.init()) {
101.    delay(1000);
102.    Serial.print("init error\r\n"); //pesan di Serial Monitor jika proses init module
103.    GPRS Gagal
104.    }
105.    Serial.println("Init succes..."); //pesan di Serial Monitor jika proses init
106.    module GPRS Sukses
107.    delay(1000);
108.    if (EEPROM[EEPROM_ADMIN_PASSWORD] == 0xFF)
109.    {
110.    EEPROM.put(EEPROM_ADMIN_PASSWORD, strAdminPassword);
```

```
111.    }
112.    Else
113.    {
114.    EEPROM.get(EEPROM_ADMIN_PASSWORD, strAdminPasswo
115.    }
116.    if (EEPROM[EEPROM_USER_PASSWORD] == 0xFF)
117.     {
118.    EEPROM.put(EEPROM_USER_PASSWORD, strUserPassword);
119.     }
120.    Else
121.    {
122.    EEPROM.get(EEPROM_USER_PASSWORD, strUserPassword);
123.    }
124.    if (EEPROM[EEPROM_PHONE_NUMBER] == 0xFF)
125.    {
126.     EEPROM.put(EEPROM_PHONE_NUMBER, strPhoneNumber);
127.    }
128.     Else
129.     {
130.    EEPROM.get(EEPROM_PHONE_NUMBER, strPhoneNumber);
131.    }
132.     kpd.attachFunction(pollingCheck);
133.    lcd.init();
134.    lcd.backlight();
135.    PORTD |= bit(DOOR_CLOSE_SWITCH_PIN)
136.    bit(DOOR_UNLOCK_SWITCH_PIN); // sets PD2 & PD3 as input pull-up
137.    bitSet(DDRD, DOOR_LOCKER_PIN); // sets PD4 as output
138.     EICRA |= bit(ISC00) | bit(ISC10); // INT0 & INT1 set on change state
139.    lcd.print(F("Door "));
140.    if (bitRead(PIND, DOOR_CLOSE_SWITCH_PIN))
141.    {
142.     lcd.print(F("opened..."));
143.    doorState = DOOR_STATE_OPENED;
144.    enableInt0();
145.    doorOpenedTone();
146.    enableSleepLCD();
147.    }
```

```
148.  Else
149.  {
150.  lcd.print(F("locked..."));
151.  doorState = DOOR_STATE_LOCKED;
152.  enableInt0();
153.  enableInt1();
154.  doorLockedTone();
155.  }
156.  }
157.  void loop()
158.  {
159.  if (isWaitingForSleep && millis() >= sleepTimeout)
160.  {
161.  sleepLCD();
162.  }
163.  if (doorState == DOOR_STATE_LOCKED)
164.  {
165.   if (getPassword())
166.  {
167.   unlockDoor();
168.  }
169.  Else
170.  {
171.  if (!kpd.isCanceled)
172.  {
173.  lcd.clear();
174.   lcd.print(F("Wrong password!!"));
175.   byte ctr = 600;
176.  while (ctr > 0 && !isInt0 && !isInt1)
177.  {
178.  doorBreakingTone();
179.  ctr--;
180.  }
181.  }
182.   if (kpd.isCanceled && !isInt0 && !isInt1)
183.  {
184.  sleepLCD();
```

```
185.    }
186.    if (isInt0)
187.    {
188.    disableInt0();
189.    if (!ensureSwitchState(DOOR_CLOSE_SWITCH_PIN))
190.    {
191.    enableInt0();
192.    return;
193.    }
194.    disableInt1();
195.    lcd.clear();
196.    lcd.print(F("Door breaking"));
197.    lcd.setCursor(0, 1);
198.    lcd.print(F("detected!!!"));
199.    gprs.sendSMS("085395262767","Pelanggaran Akses");
200.    while (true)
201.    doorBreakingTone();
202.    }
203.    }
204.    if (isInt1)
205.    {
206.    disableInt1();
207.    if (ensureSwitchState(DOOR_UNLOCK_SWITCH_PIN))
208.    {
209.    enableInt1();
210.    return;
211.    }
212.    unlockDoor();
213.    }
214.    }
215.    }
216.    else if (doorState == DOOR_STATE_UNLOCKED)
217.    {
218.    if (millis() >= unlockedTimeout)
219.    {
220.    bitClear(PORTD, DOOR_LOCKER_PIN);
221.    doorState = DOOR_STATE_LOCKED;
```

```
222.    enableInt1();
223.    lcd.clear();
224.    lcd.print(F("Door locked..."));
225.    doorLockedTone();
226.    enableSleepLCD();
227.    return;
228.    }
229.    if (isInt0)
230.    {
231.    disableInt0();
232.    if (!ensureSwitchState(DOOR_CLOSE_SWITCH_PIN))
233.    {
234.    enableInt0();
235.    return;
236.    }
237.    bitClear(PORTD, DOOR_LOCKER_PIN);
238.    doorState = DOOR_STATE_OPENED;
239.    enableInt0();
240.    lcd.clear();
241.    lcd.print(F("Door opened..."));
242.    doorOpenedTone();
243.    enableSleepLCD();
244.    }
245.    }
246.    else if (doorState == DOOR_STATE_OPENED)
247.    {
248.    if (isInt0)
249.    {
250.    disableInt0();
251.    if (ensureSwitchState(DOOR_CLOSE_SWITCH_PIN))
252.    {
253.    enableInt0();
254.    return;
255.    }
256.    doorState = DOOR_STATE_LOCKED;
257.    enableInt0();
258.    enableInt1();
```

```
259.    lcd.clear();
260.    lcd.print(F("Door locked..."));
261.    doorLockedTone();
262.    enableSleepLCD();
263.    }
264.    }
265.    }
266.    void unlockDoor()
267.    {
268.    bitSet(PORTD, DOOR_LOCKER_PIN);
269.    doorState = DOOR_STATE_UNLOCKED;
270.    unlockedTimeout = millis() + UNLOCKED_PERIOD;
271.    lcd.clear();
272.    lcd.print(F("Door unlocked..."));
273.    doorUnlockedTone();
274.    isWaitingForSleep = false;
275.    }
276.    void pollingCheck()
277.    {
278.    if (isInt0 || isInt1)
279.    {
280.    kpd.isCanceled = true;
281.    }
282.    if (isWaitingForSleep && millis() >= sleepTimeout)
283.    {
284.    kpd.isCanceled = true;
285.    }
286.    }
287.    ISR(INT0_vect)
288.    {
289.    isInt0 = true;
290.    }
291.    ISR(INT1_vect)
292.    {
293.    isInt1 = true;
294.    }
295.    void enableInt0()
```

```
296.  {
297.  cli();
298.  bitSet(EIFR, INTF0); // clears any outstanding INT0 interrupt
299.  bitSet(EIMSK, INT0); // enables INT0 interrupt
300.  sei();
301.  }
302.  void disableInt0()
303.  {
304.  cli();
305.  bitClear(EIMSK, INT0); // disables INT0 interrupt
306.  sei();
307.  isInt0 = false;
308.  }
309.  void enableInt1()
310.  {
311.  cli();
312.  bitSet(EIFR, INTF1); // clears any outstanding INT1 interrupt
313.  bitSet(EIMSK, INT1); // enables INT1 interrupt
314.  sei();
315.  }
316.  void disableInt1()
317.  {
318.  cli();
319.  bitClear(EIMSK, INT1); // disables INT1 interrupt
320.  sei();
321.  isInt1 = false;
322.  }
323.  void enableSleepLCD()
324.  {
325.  sleepTimeout = millis() + SLEEP_PERIOD;
326.   isWaitingForSleep = true;
327.  }
328.  void sleepLCD()
329.  {
330.  lcd.noBacklight();
331.   lcd.clear();
332.   lcd.noCursor();
```

```
lcd.noBlink();
isWaitingForSleep = false;
while (true)
{
key = kpd.getKey();
if (key.state == KEY_UP || key.state == CANCELED)
{
break;
}
}
lcd.backlight();
enableSleepLCD();
}
boolean ensureSwitchState(byte switchPin)
{
boolean state0 = bitRead(PIND, switchPin);
boolean state1;
while (true)
{
delay(50);
state1 = bitRead(PIND, switchPin);
if (state0 == state1);
{
return state0;
}
state0 = state1;
}
}
void doorBreakingTone()
{
float sinVal;
int toneVal;
for (byte i = 0; i < 180; i++)
{
sinVal = sin(i * 3.14159 / 180);
toneVal = 2000 + int(sinVal * 2000);
tone(BUZZER_PIN, toneVal);
```

```
370.    delay(2);
371.    if (i == 90)
372.    {
373.    lcd.noBacklight();
374.    }
375.    }
376.    noTone(BUZZER_PIN);
377.    lcd.backlight();
378.    }
379.    void doorOpenedTone()
380.    {
381.     for (int i = 1700; i < 1944; i *= 1.01)
382.    {
383.    tone(BUZZER_PIN, i);
384.    delay(30);
385.    }
386.    noTone(BUZZER_PIN);
387.    delay(100);
388.    for (int i = 1944; i > 1808; i *= 0.99)
389.    {
390.    tone(BUZZER_PIN, i);
391.    delay(30);
392.    }
393.    noTone(BUZZER_PIN);
394.    }
395.    void doorLockedTone()
396.    {
397.    for (int i = 1000; i < 2000; i *= 1.02)
398.    {
399.    tone(BUZZER_PIN, i);
400.    delay(10);
401.    }
402.    for (int i = 2000; i > 1000; i *= 0.98)
403.    {
404.    tone(BUZZER_PIN, i);
405.    delay(10);
406.    }
```

```
407.    noTone(BUZZER_PIN);
408.    }void doorUnlockedTone()
409.    {
410.    tone(BUZZER_PIN, 1568);
411.    delay(200);
412.    tone(BUZZER_PIN, 1318);
413.    delay(200);
414.    tone(BUZZER_PIN, 1046, 200);
415.    }
416.    void wrongPasswordTone()
417.    {
418.    tone(BUZZER_PIN, 1046);
419.    delay(200);
420.    tone(BUZZER_PIN, 1318);
421.    delay(200);
422.    tone(BUZZER_PIN, 1568, 200);
423.    }
424.    boolean getPassword()
425.    {
426.     enableSleepLCD();
427.    byte ctr = 3;
428.    while (ctr > 0)
429.    {
430.    lcd.clear();
431.    lcd.print(F("Password? #"));
432.    lcd.print(ctr);
433.     lcd.setCursor(0, 1);
434.    if (!getString() && kpd.isCanceled)
435.    {
436.     return false;}
437.    if (key.character == 'C')
438.    {
439.     continue;
440.    }
441.    if (strcmp(strUserPassword, strBuffer) == 0)
442.    {
443.    return true;
```

```
444.    }
445.    if (strcmp(strAdminPassword, strBuffer) == 0)
446.    {
447.    if (adminAccess())
448.    {
449.    return true;
450.    }
451.    if (kpd.isCanceled)
452.    {
453.    return false;
454.    }
455.    ctr = 3;
456.    continue;
457.    }
458.    ctr--;
459.    lcd.clear();
460.    lcd.print(F("Wrong password!"));
461.    if (ctr > 0 && !getAKey())
462.    {
463.    return false;
464.    }
465.    }
466.    return false;
467.    }
468.    boolean adminAccess()
469.    {
470.    sub_menu_1:
471.    lcd.clear();
472.    lcd.print("1.Unlock");
473.    lcd.setCursor(0, 1);
474.    lcd.print("2.Setting");
475.    while (true)
476.    {
477.    return false;
478.    }
479.    if (key.character == '1' || key.character == '2' || key.character == 'B' ||
480.    key.character == 'C')
```

```
{
tone(BUZZER_PIN, 5000, 20);
break;
}
Else
{
tone(BUZZER_PIN, 4000, 100);
}
}
if (key.character == 'B' || key.character == 'C')
{
return false;
}
else if (key.character == '1')
{
return true;
}
sub_menu_2:
lcd.clear();
lcd.print("1.Change number");
lcd.setCursor(0, 1);
lcd.print("2.Change pswd");
while (true)
{
if (!getAKey())
{
return false;
}
if (key.character == '1' || key.character == '2' || key.character == 'B' ||
key.character ==C')
{
tone(BUZZER_PIN, 5000, 20);
break;
}
Else
{
tone(BUZZER_PIN, 4000, 100);
```

```
518.    }
519.    }
520.    if (key.character == 'B')
521.    {
522.    goto sub_menu_1;
523.    }
524.    else if (key.character == 'C')
525.    {
526.    return false;
527.    }
528.    else if (key.character == '1')
529.    {
530.    if(!changeString(strPhoneNumber) && key.character == 'B')
531.    {
532.    goto sub_menu_2;
533.    }
534.    return false;
535.    }
536.    sub_menu_3:
537.    lcd.clear();
538.    lcd.print("1.Admin pswd");
539.    lcd.setCursor(0, 1);
540.    lcd.print("2.User pswd");
541.    while (true)
542.    {
543.    if (!getAKey())
544.    {
545.    return false;
546.    }
547.    if (key.character == '1' || key.character == '2' || key.character == 'B' ||
548.    key.character == 'C')
549.    {
550.    tone(BUZZER_PIN, 5000, 20);
551.    break;
552.    }
553.    Else
554.    {
```

```
555.    tone(BUZZER_PIN, 4000, 100);
556.    }
557.    }
558.    if (key.character == 'B')
559.    {
560.    goto sub_menu_2;
561.    }
562.    else if (key.character == 'C')
563.    {
564.    return false;
565.    }
566.    else if (key.character == '1')
567.    {
568.    if (!changeString(strAdminPassword) && key.character == 'B')
569.    {
570.    goto sub_menu_3;
571.    }
572.    return false;
573.    }
574.    if (!changeString(strUserPassword) && key.character == 'B')
575.    {
576.    goto sub_menu_3;
577.    }
578.    return false;
579.    }
580.    boolean changeString(char *strDestination)
581.    {
582.    boolean oldIsAlphaMode = isAlphaMode;
583.    boolean oldIsUpperCaseMode = isUpperCaseMode;
584.    lcd.clear();
585.    if (strDestination == strPhoneNumber)
586.    {
587.    lcd.print(F("Current number:"));
588.    }
589.    else if (strDestination == strAdminPassword)
590.    {
591.    lcd.print(F("Cur admin pswd:"));
```

```
592.    }
593.    else if (strDestination == strUserPassword)
594.    {
595.    lcd.print(F("Cur user pswd:"));
596.    }
597.    lcd.setCursor(0, 1);
598.    lcd.print(strDestination);
599.    if (!getAKey())
600.    {
601.    return false;
602.    }
603.    if (key.character == 'B' || key.character == 'C')
604.    {
605.    tone(BUZZER_PIN, 5000, 20);
606.    return false;
607.    }
608.    lcd.clear();
609.    if (strDestination == strPhoneNumber)
610.    {
611.    lcd.print(F("New number?"));
612.    isAlphaMode = false;
613.    }
614.    else if (strDestination == strAdminPassword)
615.    {
616.      lcd.print(F("New adm pwd?"));
617.     isAlphaMode = true;
618.    }
619.    else if (strDestination == strUserPassword)
620.    {
621.    lcd.print(F("New usr pwd?"));
622.     isAlphaMode = true;
623.    }
624.    boolean gs = getString();
625.    isAlphaMode = oldIsAlphaMode;
626.    isUpperCaseMode = oldIsUpperCaseMode;
627.     if (!gs || key.character == 'C')
628.    {
```

```
629.  return false;
630.  }
631.  strcpy(strDestination, strBuffer);
632.  lcd.clear();
633.  if (strDestination == strPhoneNumber)
634.  {
635.  if (!saveToEEPROM(EEPROM_PHONE_NUMBER, strDestination))
636.  {
637.  return false;
638.  }
639.  lcd.print(F("Phone number"));
640.  }
641.  else if (strDestination == strAdminPassword)
642.  {
643.   if (!saveToEEPROM(EEPROM_ADMIN_PASSWORD, strDestination))
644.  {
645.  return false;
646.  }
647.  lcd.print(F("Admin password"));
648.  }
649.  else if (strDestination == strUserPassword)
650.  {
651.  if (!saveToEEPROM(EEPROM_USER_PASSWORD, strDestination))
652.  {
653.  return false;
654.  }
655.  lcd.print(F("User password"));
656.  }
657.  lcd.setCursor(0, 1);
658.  lcd.print(F("changed..."));
659.  getAKey();
660.  return true;
661.  }
662.  boolean saveToEEPROM(int address, char *str)
663.  {
664.  //EEPROM.put(address, str);
665.  // ZulNs: I don't know why above
```

```
666.    // code didn't working, instead
667.    // I use following code which
668.    // write one by one byte to EEPROM.
669.    for (byte i = 0; i < 16; i++)
670.    {
671.    EEPROM.update(address + i, str[i]);
672.    }
673.     EEPROM.get(address, strBuffer);
674.    if (strcmp(str, strBuffer) != 0)
675.    {
676.    lcd.print(F("Failed to save"));
677.    lcd.setCursor(0, 1);
678.    lcd.print(F("to EEPROM!!!"));
679.     byte ctr = 30;
680.    while (ctr > 0 && !isInt0 && !isInt1)
681.    {
682.    doorBreakingTone();
683.    ctr--;
684.    }
685.    lcd.clear();
686.    enableSleepLCD();
687.    return false;
688.    }
689.    return true;
690.    }
691.    void getKeyObject()
692.    {
693.     key = kpd.getKey();
694.    if (key.state == KEY_UP)
695.    {
696.    enableSleepLCD();
697.    }
698.    Else
699.    {
700.    isWaitingForSleep = false;
701.    }
702.    }
```

```
boolean getAKey()
{
while(true)
{
getKeyObject();
if (kpd.isCanceled)
{
return false;
}
if (key.state == KEY_UP && key.character != 0)
{
return true;
}
}
}
boolean getString()
{
lcd.cursor();
lcd.blink();
boolean result = getStringPD(16, 30);
lcd.noCursor();
lcd.noBlink();
return result;
}
boolean getStringPD(byte startPos, byte endPos)
{
char chr;
byte strSize = endPos - startPos + 1;
startCursorPos = startPos;
endCursorPos = endPos;
cursorPos = startPos;
setCursorPos();
chrCtr = 0;
displayInputMode();
while (true)
{
getKeyObject();
```

```
740.    chr = key.character;
741.    switch (key.state)
742.    {
743.    case CANCELED:
744.    return false;
745.    case KEY_DOWN:
746.    case MULTI_TAP:
747.    tone(BUZZER_PIN, 5000, 20);
748.    digitalWrite(LED_BUILTIN, HIGH);
749.    switch (key.code)
750.    {
751.    case KEY_1:
752.    if (isAlphaMode)
753.    {
754.    chr = getSymbol(key.tapCounter, SYMBOL_1);
755.    }
756.    break;
757.    case KEY_0:
758.    if (isAlphaMode)
759.    {
760.    chr = ' ';
761.    }
762.    break;
763.    case KEY_ASTERISK:
764.    if (isAlphaMode)
765.    {
766.    chr = getSymbol(key.tapCounter, SYMBOL_A);
767.    }
768.    break;
769.    case KEY_NUMBER_SIGN:
770.    if (isAlphaMode)
771.    {
772.    chr = getSymbol(key.tapCounter, SYMBOL_NS);
773.    }
774.    break;
775.    if (isAlphaMode)
776.    {
```

```
777.    isUpperCaseMode = !isUpperCaseMode;
778.    chr = isUpperCaseMode ? CAPSLOCK_ON : CAPSLOCK_OFF;
779.    }
780.    Else
781.    {
782.    chr = 0;
783.    }
784.    break;
785.    case KEY_B:
786.    chr = BACKSPACE;
787.    break;
788.    case KEY_C:
789.    chrCtr = 0;
790.    strBuffer[0] = 0;
791.    chr = CLEARSCREEN;
792.    break;
793.    case KEY_D:
794.    strBuffer[chrCtr] = 0;
795.    chr = CARRIAGE_RETURN;
796.    break;
797.    default:
798.    if (isAlphaMode)
799.    {
800.    chr = getAlphabet(key.character, key.tapCounter);
801.    if (!isUpperCaseMode)
802.    {
803.    chr += 32;  // makes lower case
804.    }
805.    }
806.    }
807.    if (key.state == MULTI_TAP && isAlphaMode && key.character < 'A' &&
808.    key.character != '0')
809.    {
810.    printToLcd(BACKSPACE);
811.    chrCtr--;
812.    }
813.    if (chr == BACKSPACE)
```

```
{
chrCtr--;
}
if (chr >= ' ')
{
strBuffer[chrCtr] = chr;
if (chrCtr < strSize)
{
chrCtr++;
}
}
if (chr != CARRIAGE_RETURN && chr !=0)
{
printToLcd(chr);
}
break;
case LONG_TAP:
switch (key.code)
{
case KEY_A:
if (isAlphaMode)
{
isUpperCaseMode = !isUpperCaseMode;
isAlphaMode = false;
chr = NUMLOCK_ON;
}
Else
{
isAlphaMode = true;
chr = NUMLOCK_OFF;
}
break;
case KEY_B:
chr = CLEARSCREEN;
chrCtr = 0;
break;
case KEY_C:
```

```
851.    case KEY_D:
852.    chr = 0;
853.    break;
854.    default:
855.    if (!isAlphaMode)
856.    {
857.    chr = 0;
858.    }
859.    }
860.    if (chr > 0)
861.    {
862.    tone(BUZZER_PIN, 5000, 20);
863.    if (' ' < chr && chr < 'A')
864.    {
865.    printToLcd(BACKSPACE);
866.    chrCtr--;
867.    strBuffer[chrCtr] = chr;
868.     if (chrCtr < strSize)
869.    {
870.    chrCtr++;
871.    }
872.    }
873.    printToLcd(chr);
874.    }
875.    break;
876.    case MULTI_KEY_DOWN:
877.    tone(BUZZER_PIN, 4000, 100);
878.    break;
879.    case KEY_UP:
880.    if (key.character == 'D' && chrCtr > 0)
881.    {
882.    return true;
883.    }
884.    if (key.character == 'C')
885.    {
886.    return false;
887.    }
```

```
888.    }
889.    }
890.    }
891.    void printToLcd(char chr)
892.    {
893.    switch (chr)
894.    {
895.    case BACKSPACE:
896.    if (cursorPos > startCursorPos)
897.    {
898.    if (!isEndOfDisplay)
899.    {
900.    decCursorPos();
901.    }
902.    setCursorPos();
903.    lcd.print(F(" "));
904.    setCursorPos();
905.    isEndOfDisplay = false;
906.    }
907.    break;
908.    case CLEARSCREEN:
909.    while (cursorPos > startCursorPos)
910.    {
911.    decCursorPos();
912.    setCursorPos();
913.    lcd.print(F(" "));
914.    setCursorPos();
915.    }
916.    break;
917.    case CAPSLOCK_ON:
918.    case CAPSLOCK_OFF:
919.    case NUMLOCK_ON:
920.    case NUMLOCK_OFF:
921.    displayInputMode();
922.    break;
923.    default:
924.    if (cursorPos == endCursorPos)
```

```
{
isEnndOfDisplay = true;
}
lcd.print(chr);
incCursorPos();
setCursorPos();
}
}
void displayInputMode()
{
lcd.setCursor(13, 0);
if (isAlphaMode)
{
if (isUpperCaseMode)
{
lcd.print(F("ABC"));
}
Else
{
lcd.print(F("abc"));
}
}
Else
{
lcd.print(F("123"));
}
setCursorPos();
}
void incCursorPos()
{
if (cursorPos < endCursorPos)
{
cursorPos=++;
}
}
void decCursorPos()
{
```

```
962.  if (cursorPos > startCursorPos)
963.  {
964.  cursorPos--;
965.  }
966.  }
967.  void setCursorPos()
968.  {
969.  lcd.setCursor(cursorPos % 16, cursorPos / 16);
970.  }
971.  byte getSymbol(byte ctr, char * pointer)
972.  {
973.  byte chr = pgm_read_byte_near(pointer + ctr);
974.  if (chr == CHR_BOUND)
975.  {
976.  chr = pgm_read_byte_near(pointer);
977.  kpd.resetTapCounter();
978.  }
979.  return chr;
980.  }
981.  byte getAlphabet(byte chr, byte ctr)
982.  {
983.  chr = (chr - '2') * 5;
984.  byte alpha = pgm_read_byte_near(ALPHABET + chr + ctr);
985.  if (alpha == CHR_BOUND)
986.  {
987.  alpha = pgm_read_byte_near(ALPHABET + chr);
988.  kpd.resetTapCounter();
989.  }
990.  return alpha;
991.  }
```

# CURRICULUM VITAE

## 1. Indentitas

**ABDULLAH DANIAL,** Lahir di Desa helumo 11 September 1992. Merupakan putra pertama dari empat bersaudara pasangan Amir Danial dan Fatma Ahmad. Menjadi mahasiswa Strata S1 Teknik Elektro Fakultas Teknik Universitas Negeri Gorontalo angkatan 2012. Berdomisili Desa pilomonu kecamatan mootilango kabupaten Gorontalo.

## 2. Riwayat Pendidikan

- Pemdidikan Formal
  1. SDN Negeri 6 Mootilango
  2. SMP Negeri 2 Mootilango
  3. SMK Negeri 1 Batudaa
  4. Teknik Elektro Fakultas Teknik Univesitas Negeri Gorontalo Tahun 2012.

- Pemdidikan Non Formal
  1. Anak kader Aliansi Mahasiswa Pelajar Kawasan Paguyaman Raya tahun 2012 (AMPKPR)
  2. Masa Orientasi Mahasiswa Baru tahun 2012 (MOMB)
  3. Anak kader Ikatan Pelajar Mahasiswa Bolihuto Raya tahun 2012 (IPMB).
  4. Peserta Kuliah Kerja Sibermas (KKS) Universitas Negeri Gorontalo tahun 2016.
  5. Peserta Kerja Praktek Fakutas Teknik tahun 2017 di PT Telkom Indosia
  6. Peserta pelatihan Instalasi Listrik UPTD BLKLT PROV.GORONTALO.
  7. Pengurus REMAJA MASJID desa pilomonu KEC. MOOTILANGO tahun 2017/2018