BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengujian Sistem

Pengujian sistem disini berdasarkan rancangan desain yang telah dibuat dan untuk mengetahui apakah sistem yang telah dibuat dapat berfungsi dengan sesuai yang diharapkan atau tidak. Berikut adalah uji coba sistem dari pengerjaan sistem berdasarkan rancangan desain yang telah dibuat. Adapun implementasinya pada Kotak Penyimpanan Uang (*MoneyBos Plus*) sebagai berikut :

4.1.1 Menu utama

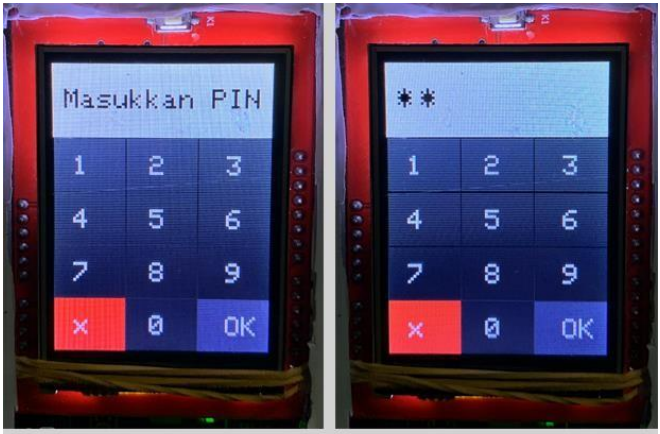
Uji coba pada menu utama ini merupakan akses yang akan diguanakan sebagai tampilan awal yang nantinya bisa menjadi jembatan kepada fitur scan uang yang ada pada Kotak Penyimpanan Uang (*MoneyBox Plus*).

Gambar 4. 1 Menu Utama

4.1.2 Menu Cek Jumlah

Menu Cek Jumlah merupakan akses yang di gunakan untuk melihat jumlah uang pada Kotak Penyimpanan Uang (*MoneyBox Plus*) dan hanya orang berhak saja yang memiliki akses. Adapun Pengujian Menu Akses Masuk ini adalah sebagai berikut.

1. Setelah masuk ke Menu Akses Masuk, masukkan PIN atau kode pengaman yang sesuai dengan PIN atau kode pengaman pada sistem untuk masuk ke dalam akses Kotak Penyimpanan Uang (*MoneyBox Plus*).



Gambar 4. 2 UI Menu Cek Jumlah

2. Jika PIN atau kode pengaman yang dimasukkan sesuai dengan PIN atau kode pengaman pada sistem, maka akan muncul jumlah uang yang telah tersimpan pada Kotak Penyimpanan Uang (*MoneyBox Plus*).

Gambar 4. 3 UI menampilkan jumlah uang

3. Sedangkan jika PIN atau kode pengaman yang dimasukkan tidak sesuai dengan PIN atau kode pengaman pada sistem, maka akses ditolak dan Kotak Penyimpanan Uang (*MoneyBox Plus*) tetap terkunci.

Gambar 4. 4 Akses Cek Jumlah Ditolak

4.1.3 Penerapan Cara Kerja Sistem Sortir

Penerapan cara kerja deteksi warna pada Kotak Penyimpanan Uang (*MoneyBox Plus*) sebagai berikut :

1. Sambungkan alat *MoneyBox Plus* pada *Power Supply*, yang dimana alat yang digunakan sebagai *power supply* ini menggunakan kabel data dari USB.
2. Setelah alat terhubung dengan *power supply* maka arduino yang bekerja sebagai otak mulai bekerja untuk menjalankan mesin pada Kotak Penyimpanan Uang (*MoneyBox Plus*).

3. Selanjutanya, menu utama dari Kotak Penyimpanan Uang ini akan menyala.
4. Tekan tombol pada layar dibagian Scan Uang, dan nanti akan pindah ke bagian Scan Uang.
5. Masukkan uang pecahan Rp.50.000 atau Rp.100.000 yang mau ditaruh ke Kotak Penyimpanan Uang (*MoneyBox Plus*), batas memasukkan uang dalam satu fase adalah 5 kali memasukkan uang setelah itu Kotak Penyimpanan Uang (*MoneyBox Plus*) akan menampilkan total uang yang sudah discan melalui scan warna.
6. Setelah memasukan uang pecahan Rp. 50.000 atau Rp.100.000 *Motor Servo* yang ada pada Kotak Penyimpanan uang (*MoneyBox Plus)* akan ke-*Trigger.*
7. Jika *Motor Servo* sudah ke-*Trigger* maka akan bergerak untuk membuka kotak pemisah yang ada di dalam Kotak Penyimpanan uang (*MoneyBox Plus).*
8. Setelah itu akan terbuka masing-masing kotak yang ada pada Kotak Penyimpanan uang (*MoneyBox Plus)* yang dimana akan mensortir uang pecahan Rp.50.000 atau Rp.100.000 sesuai kotak yang telah tersedia di dalam Kotak Penyimpanan uang (*MoneyBox Plus).*

4.2 Hasil Penelitian

Berdasarkan pengujian sistem yang dilakukan saat penelitian terhadap kotak Penyimpanan Uang (*MoneyBos Plus*) memperoleh hasil sebagai berikut :

4.2.1 *Layout* Kode Menu Utama

```
324  void Menu_display()
325  {
326    tft.setTextSize(3);
327    tft.setTextColor(BLACK);
328    tft.setCursor(40, 37);
329    tft.print("MoneyBox");
330
331    tft.setTextSize(3);
332    tft.setTextColor(BLACK);
333    tft.setCursor(185, 22);
334    tft.print("+");
335
336    DrawButtonScanUang();
337    DrawButtonCekJumlah();
338    DrawButtonAksesMasuk();
339    DrawButtonResetPIN();
340    DrawButtonLupaPIN();
341    DrawButtonClear();
342  }
```

Gambar 4. 5 *Source Code* Menu Utama

Pada gambar diatas, penjelasan mengenai kode dalam fungsi "Menu *Display*" pada program Arduino adalah sebagai berikut:

1. `tft.*setTextSize*(3);`: Mengatur ukuran teks yang akan ditampilkan pada layar menjadi 3 kali ukuran default.
2. `tft.*setTextColor*(BLACK);`: Mengatur warna teks menjadi hitam.
3. `tft.*setCursor*(40, 37);`: Mengatur posisi kursor teks pada layar dengan koordinat (40, 37). Koordinat ini menentukan letak awal teks yang akan ditampilkan.
4. `tft.print("MoneyBox");`: Mencetak teks "MoneyBox" pada layar pada posisi yang telah ditentukan sebelumnya.
5. `tft.*setTextSize*(3);`: Mengatur ukuran teks kembali menjadi 3 kali ukuran default.
6. `tft.*setTextColor*(BLACK);`: Mengatur warna teks kembali menjadi hitam.
7. `tft.*setCursor*(185, 22);`: Mengatur posisi kursor teks pada layar dengan koordinat (185, 22).
8. `tft.print("+");`: Mencetak tanda "+" pada layar pada posisi yang telah ditentukan sebelumnya.
9. Selanjutnya, terdapat pemanggilan beberapa fungsi lain, seperti `*DrawButtonScan*Uang()`, `*DrawButton*CekJumlah()`, `*DrawButton*AksesMasuk()`, `*DrawButtonReset*PIN()`, `*DrawButtonLupa*PIN()`, dan `*DrawButtonClear*()`. Fungsi-fungsi ini bertanggung jawab untuk menggambar dan menampilkan tombol-tombol pada layar.

Dengan menjalankan fungsi "*Menu Display*", layar akan menampilkan teks "*MoneyBox*" dan beberapa tombol yang tergambar sesuai dengan pemanggilan fungsi-fungsi yang ada di dalamnya. Fungsi ini bertanggung jawab untuk menampilkan tampilan menu utama pada layar TFT (*Thin-Film Transistor*) display pada Arduino.

4.2.2 Layout Kode Menu Cek Jumlah

```
void DrawButtonCekJumlah()
{
  tft.fillRoundRect(123, 95, 95, 70, 10, BLACKM);
  tft.fillRoundRect(125, 97, 91, 66, 10, BLUE);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(153, 110);
  tft.print("CEK");
  tft.setCursor(135, 135);
  tft.print("JUMLAH");
}
```

Gambar 4. 6 Source Code Menu Cek Jumlah Fungsi 1

Pada gambar diatas merupakan fungsi pertama, DrawButtonCekJumlah(), digunakan untuk menggambar tombol dengan teks "CEK JUMLAH". Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. `tft.*fillRoundRect*(123, 95, 95, 70, 10, BLACKM);`: Fungsi ini menggambar sebuah persegi panjang dengan sudut yang dibulatkan pada posisi (123, 95) dengan lebar 95 piksel dan tinggi 70 piksel. Parameter terakhir, `BLACKM`, kemungkinan merujuk pada warna hitam dalam format yang digunakan oleh library yang digunakan dalam proyek ini.

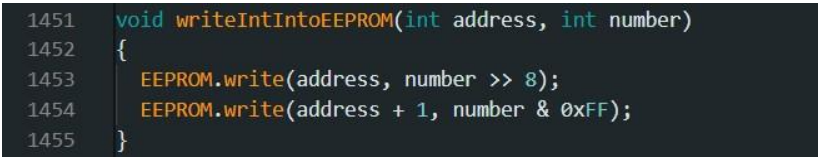
2. `tft.*fillRoundRect*(125, 97, 91, 66, 10, BLUE);`: Fungsi ini menggambar persegi panjang dengan sudut yang dibulatkan pada posisi (125, 97) dengan lebar 91 piksel dan tinggi 66 piksel. Perbedaannya adalah warna yang digunakan kali ini adalah biru (BLUE).
3. `tft.*setTextSize*(2);`: Fungsi ini mengatur ukuran teks pada layar menjadi 2.
4. `tft.*setTextColor*(WHITE);`: Fungsi ini mengatur warna teks menjadi putih (WHITE).
5. `tft.*setCursor*(153, 110);`: Fungsi ini menentukan posisi kursor untuk penulisan teks berikutnya. Dalam hal ini, kursor ditempatkan pada posisi (153, 110).
6. `tft.print("CEK");`: Fungsi ini mencetak teks "CEK" pada posisi kursor yang telah ditentukan sebelumnya.
7. `tft.*setCursor*(135, 135);`: Fungsi ini menentukan posisi kursor untuk penulisan teks selanjutnya. Dalam hal ini, kursor ditempatkan pada posisi (135, 135).
8. `tft.print("JUMLAH");`: Fungsi ini mencetak teks "JUMLAH" pada posisi kursor yang baru ditentukan.

```
373  void DrawButtonCekJumlahPress()
374  {
375    tft.fillRoundRect(123, 95, 95, 70, 10, BLACKM);
376  }
```

Gambar 4. 7 *Source Code* Menu Cek Jumlah Fungsi 2

Pada gambar di atas merupakan Fungsi `*DrawButton*CekJumlahPress(). Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. `tft.*fillRoundRect*(123, 95, 95, 70, 10, BLACKM);`: Fungsi ini menggambar persegi panjang dengan sudut yang dibulatkan pada posisi (123, 95) dengan lebar 95 piksel dan tinggi 70 piksel. Perbedaannya adalah kali ini persegi panjang diisi dengan warna hitam (BLACKM), menciptakan efek tekanan pada tombol.



```
1451  void writeIntIntoEEPROM(int address, int number)
1452  {
1453    EEPROM.write(address, number >> 8);
1454    EEPROM.write(address + 1, number & 0xFF);
1455  }
```

Gambar 4. 8 *Source Code* Menu Cek Jumlah Fungsi 3

Pada gambar diatas merupakan Fungsi *write*IntIntoEEPROM(int *address*, int *number*) Fungsi ini digunakan untuk menulis sebuah bilangan bulat (*integer*) ke dalam EEPROM (*Electrically Erasable Programmable Read-Only Memory*). EEPROM adalah jenis memori yang dapat menyimpan data bahkan ketika daya listrik terputus. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. EEPROM.*write*(address, number >> 8);: Pada baris ini, kita menggunakan fungsi EEPROM.*write*() untuk menulis data ke alamat yang ditentukan dalam EEPROM. Parameter pertama, address, adalah alamat memori EEPROM di mana kita ingin menyimpan bilangan tersebut. Parameter kedua, number >> 8, adalah bilangan yang akan ditulis ke EEPROM setelah digeser 8 bit ke kanan. Hal ini dilakukan untuk mendapatkan 8 bit paling signifikan (most significant bits/MSB) dari bilangan number.

2. EEPROM.*write*(address + 1, number & 0xFF);: Pada baris ini, kita menggunakan fungsi EEPROM.*write*() sekali lagi untuk menulis data ke alamat yang berikutnya dalam EEPROM. Alamat yang digunakan adalah address + 1, untuk menulis 8 bit paling tidak signifikan (least significant bits/LSB) dari bilangan number. number & 0xFF mengambil 8 bit terakhir dari number dengan menggunakan operasi bitwise AND dengan 0xFF.

Gambar 4. 9 *Source Code* Menu Cek Jumlah Fungsi 4

Pada gambar diatas merupakan fungsi readIntFromEEPROM(int address) Fungsi ini digunakan untuk membaca sebuah bilangan bulat (integer) dari EEPROM. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. (EEPROM.read(address)<< 8) + EEPROM.read(address + 1);: Baris ini membaca data dari EEPROM dengan menggunakan fungsi EEPROM.read(). Kita membaca byte pertama dari alamat yang ditentukan, EEPROM.read(address), kemudian menggeser (shift) nilai byte tersebut ke kiri (left shift) sebanyak 8 bit menggunakan operator bitwise shift (<< 8).
2. Setelah itu, kita menambahkan hasil dari pergeseran bit tersebut dengan byte kedua yang dibaca dari alamat berikutnya, EEPROM.read(address + 1). Hal ini

dilakukan untuk menggabungkan nilai byte pertama (MSB) dengan byte kedua (LSB) dan menghasilkan bilangan bulat yang akan dikembalikan oleh fungsi.

```
1462  void SimpanTotalUang()
1463  {
1464    jumlahUang = readIntFromEEPROM(23);
1465    jumlahUang = jumlahUang + totalSementara;
1466    writeIntIntoEEPROM(23, jumlahUang);
1467  }
```

Gambar 4. 10 *Source Code* Menu Cek Jumlah Fungsi 5

Pada gambar diatas merupakan fungsi SimpanTotalUang() Fungsi ini digunakan untuk menyimpan total uang ke dalam EEPROM. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. jumlahUang = *read*IntFromEEPROM(23);: Pada baris ini, kita menggunakan fungsi readIntFromEEPROM() untuk membaca nilai total uang yang telah disimpan sebelumnya dari alamat EEPROM 23. Nilai yang dibaca kemudian disimpan dalam variabel jumlahUang.
2. jumlahUang = jumlahUang + totalSementara;: Setelah itu, kita menambahkan nilai totalSementara ke dalam jumlahUang. Ini bertujuan untuk mengupdate nilai total uang dengan menambahkan nilai sementara yang baru.
3. *write*IntIntoEEPROM(23, jumlahUang);: Pada baris ini, kita menggunakan fungsi *write*IntIntoEEPROM() untuk menulis nilai yang telah diubah ke dalam alamat EEPROM 23.

```
1469  void TampilanResetTotalUang()
1470  {
1471    Serial.println("Total Uang Telah Direset");
1472    tft.fillScreen(WHITE);
1473    tft.setTextSize(3);
1474    tft.setTextColor(JADE);
1475    tft.setCursor(28, 124);
1476    tft.println("Total Uang");
1477    tft.setCursor(3, 164);
1478    tft.println("Telah Direset");
1479  }
```

Gambar 4. 11 *Source Code* Menu Cek Jumlah fungsi 6

Pada gambar diatas merupakan fungsi TampilanResetTotalUang(). Fungsi ini bertujuan untuk menampilkan pesan bahwa total uang telah direset dan melakukan beberapa pengaturan tampilan pada layar TFT (Thin-Film Transistor). Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. Serial.println("Total Uang Telah Direset");: Pada baris ini, pesan "Total Uang Telah Direset" akan dicetak ke output Serial Monitor. Ini berguna jika Anda menggunakan koneksi serial untuk melihat pesan ini dalam lingkungan pemrograman Arduino atau sejenisnya.
2. tft.*fillScreen*(WHITE);: Baris ini akan mengisi seluruh layar TFT dengan warna putih. tft adalah objek yang digunakan untuk mengendalikan tampilan pada layar TFT.
3. tft.*setTextSize*(3);: Pada baris ini, ukuran teks pada layar TFT diatur menjadi 3. Ini menentukan ukuran teks yang akan digunakan untuk teks selanjutnya yang ditampilkan pada layar.

4. tft.*setTextColor*(JADE);: Baris ini mengatur warna teks pada layar TFT menjadi warna "JADE". "JADE" mungkin merupakan konstanta yang didefinisikan sebelumnya dalam program atau kode Anda.
5. tft.*setCursor*(28, 124);: Pada baris ini, posisi kursor teks pada layar TFT diatur ke koordinat (28, 124). Ini menentukan posisi awal di mana teks akan ditampilkan pada layar.
6. tft.println("Total Uang");: Baris ini akan mencetak teks "Total Uang" pada layar TFT sesuai dengan posisi kursor yang telah diatur sebelumnya.
7. tft.*setCursor*(3, 164);: Pada baris ini, posisi kursor teks pada layar TFT diatur ke koordinat (3, 164). Ini menentukan posisi awal di mana teks selanjutnya akan ditampilkan pada layar.
8. tft.println("Telah Direset");: Baris ini akan mencetak teks "Telah Direset" pada layar TFT sesuai dengan posisi kursor yang telah diatur sebelumnya.

Dengan demikian, fungsi TampilanResetTotalUang() akan menampilkan pesan "Total Uang Telah Direset" pada output Serial Monitor (jika ada) dan melakukan pengaturan tampilan pada layar TFT untuk menampilkan teks "Total Uang" dan "Telah Direset".

4.2.3 Layout Kode *Motor Servo*

Gambar 4. 12 *Source Code Motor Servo* fungsi 1

Pada gambar diatas merupakan Source Code yang berfungsi untuk mendefinisikan dua variabel bertipe int, yaitu position1 dan position2, dan memberikan nilai awal masing-masing variabel tersebut. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. int *position*1 = 100;: Baris ini mendefinisikan variabel *position*1 dengan tipe data int (integer) dan memberikan nilai awal 100 ke variabel tersebut. Variabel position1 kemungkinan digunakan untuk menyimpan posisi yang akan diberikan ke servo yang terhubung ke servo_28.
2. int *position*2 = 140;: Baris ini mendefinisikan variabel *position*2 dengan tipe data int (integer) dan memberikan nilai awal 140 ke variabel tersebut. Variabel position2 kemungkinan digunakan untuk menyimpan posisi yang akan diberikan ke servo yang terhubung ke servo_30.

Dengan memberikan nilai awal ini dapat diatur posisi awal dari servo motor ketika program dijalankan, dan dapat mengubah nilai position1 dan position2 sesuai dengan kebutuhan untuk mengatur posisi servo motor yang diinginkan pada awal program.

```
76  Servo servo;
77  Servo servo_28;
78  Servo servo_30;
```

Gambar 4. 13 *Source Code Motor Servo* fungsi 2

Pada gambar diatas mendefinisikan tiga objek Servo dengan nama servo, servo_28, dan servo_30. Objek-objek ini digunakan untuk mengendalikan servo motor pada mikrokontroler atau papan Arduino. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. Servo servo;: Baris ini mendefinisikan objek servo dengan tipe data Servo. Objek ini digunakan untuk mengendalikan sebuah servo motor.
2. Servo servo_28;: Baris ini mendefinisikan objek servo_28 dengan tipe data Servo. Objek ini juga digunakan untuk mengendalikan sebuah servo motor.
3. Servo servo_30;: Baris ini mendefinisikan objek servo_30 dengan tipe data Servo. Objek ini juga digunakan untuk mengendalikan sebuah servo motor.

Dengan mendefinisikan objek-objek Servo ini dapat menggunakan objek-objek tersebut untuk menghubungkan servo motor ke pin tertentu pada mikrokontroler atau papan Arduino, serta menggerakkan servo ke posisi yang diinginkan menggunakan fungsi-fungsi yang disediakan oleh library Servo. Misalnya, Anda dapat menggunakan fungsi attach() untuk menghubungkan servo ke pin tertentu, dan fungsi *write*() untuk menggerakkan servo ke posisi tertentu.

```
124    servo_28.attach(28);
125    servo_30.attach(30);
126
127    servo_28.write(position1);
128    servo_30.write(position2);
```

Gambar 4. 14 *Source Code Motor Servo* fungsi 3

Pada gambar diatas merupaka Source Code yang berfungsi untuk mengontrol dua servo motor. Mari kita

jelaskan setiap barisnya. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut :

1. servo_28.attach(28);: Pada baris ini, servo dengan nama servo_28 dihubungkan ke pin digital 28 pada mikrokontroler atau papan Arduino yang digunakan. Fungsi attach() digunakan untuk menghubungkan objek servo ke pin yang ditentukan.
2. servo_30.attach(30);: Pada baris ini, servo dengan nama servo_30 dihubungkan ke pin digital 30 pada mikrokontroler atau papan Arduino yang digunakan. Fungsi attach() digunakan untuk menghubungkan objek servo ke pin yang ditentukan.
3. servo_28.*write*(position1);: Baris ini mengatur posisi servo servo_28 sesuai dengan nilai yang ada dalam variabel position1. Fungsi *write*() digunakan untuk menggerakkan servo ke posisi tertentu yang diberikan dalam parameter.
4. servo_30.*write*(position2);: Baris ini mengatur posisi servo servo_30 sesuai dengan nilai yang ada dalam variabel position2. Fungsi *write*() digunakan untuk menggerakkan servo ke posisi tertentu yang diberikan dalam parameter.

Dalam kode tersebut, servo motor dihubungkan ke pin digital 28 dan 30, dan kemudian digerakkan ke posisi yang telah ditentukan oleh variabel position1 dan position2. Anda perlu memastikan bahwa pin-pin yang digunakan sesuai dengan koneksi fisik pada mikrokontroler atau papan Arduino yang Anda gunakan, dan juga memastikan bahwa variabel position1 dan position2 berisi nilai posisi yang *valid* untuk servo motor yang digunakan.

```
if (r>48 && r<70 && g>102 && g<117 && b>86 && b<110)
{
  tft.fillRoundRect(15, 100, 210, 80, 5, BLACKM);
  tft.fillRoundRect(17, 102, 206, 76, 5, LIMAPULUH);

  tft.setCursor(40, 129);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Rp.50.000");
  limapuluh = 50;


  servo_28.write(65);
  delay(3000);
  servo_28.write(position1);
}
```

Gambar 4. 15 Source Code *Motor Servo* fungsi 4

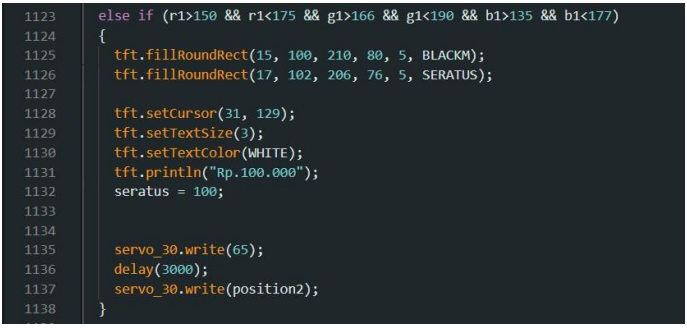
Pada gambar diatas merupakan *Source Code* adalah sebuah blok if yang melakukan pengecekan kondisi pada variabel r, g, dan b. Jika kondisi tersebut terpenuhi, maka blok kode di dalamnya akan dieksekusi. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut:

1. if (r > 48 && r < 70 && g > 102 && g < 117 && b > 86 && b < 110): Baris ini adalah kondisi yang harus terpenuhi agar blok kode di dalamnya dieksekusi. Kondisi ini memeriksa apakah nilai variabel r berada di antara 48 dan 70, nilai variabel g berada di antara 102 dan 117, serta nilai variabel b berada di antara 86 dan 110.
2. {: Kurung kurawal membuka blok kode yang akan dieksekusi jika kondisi di atas terpenuhi.
3. tft.fillRoundRect(15, 100, 210, 80, 5, BLACKM);: Baris ini mengisi suatu area persegi panjang dengan sudut

melengkung pada layar TFT menggunakan warna BLACKM. Area yang diisi dimulai dari koordinat (15, 100) dengan lebar 210 piksel dan tinggi 80 piksel.

4. tft.fillRoundRect(17, 102, 206, 76, 5, LIMAPULUH);: Baris ini mengisi suatu area persegi panjang dengan sudut melengkung pada layar TFT menggunakan warna LIMAPULUH. Area yang diisi dimulai dari koordinat (17, 102) dengan lebar 206 piksel dan tinggi 76 piksel.
5. tft.*setCursor*(40, 129);: Baris ini mengatur posisi kursor pada layar TFT ke koordinat (40, 129). Kursor ini akan digunakan untuk menampilkan teks selanjutnya pada layar.
6. tft.*setTextSize*(3);: Baris ini mengatur ukuran teks pada layar TFT menjadi 3.
7. tft.*setTextColor*(WHITE);: Baris ini mengatur warna teks pada layar TFT menjadi WHITE.
8. tft.println("Rp.50.000");: Baris ini mencetak teks "Rp.50.000" pada layar TFT sesuai dengan posisi kursor yang telah diatur sebelumnya.
9. limapuluh = 50;: Baris ini mengassign nilai 50 ke variabel limapuluh.
10. servo_28.*write*(65);: Baris ini menggerakkan servo yang terhubung ke servo_28 ke posisi 65. Fungsi *write*() digunakan untuk menggerakkan servo ke posisi tertentu yang diberikan dalam parameter.
11. delay(3000);: Baris ini menyebabkan program menghentikan eksekusi selama 3000 milidetik (3 detik).
12. servo_28.*write*(position1);: Baris ini menggerakkan servo yang terhubung ke servo_28 ke posisi yang ditentukan oleh variabel position1.

Dengan demikian, jika kondisi pada baris pertama terpenuhi, maka blok kode di dalamnya akan di eksekusi dan *Motor Servo* akan bergerak.



```
else if (r1>150 && r1<175 && g1>166 && g1<190 && b1>135 && b1<177)
{
  tft.fillRoundRect(15, 100, 210, 80, 5, BLACKM);
  tft.fillRoundRect(17, 102, 206, 76, 5, SERATUS);

  tft.setCursor(31, 129);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Rp.100.000");
  seratus = 100;


  servo_30.write(65);
  delay(3000);
  servo_30.write(position2);
}
```

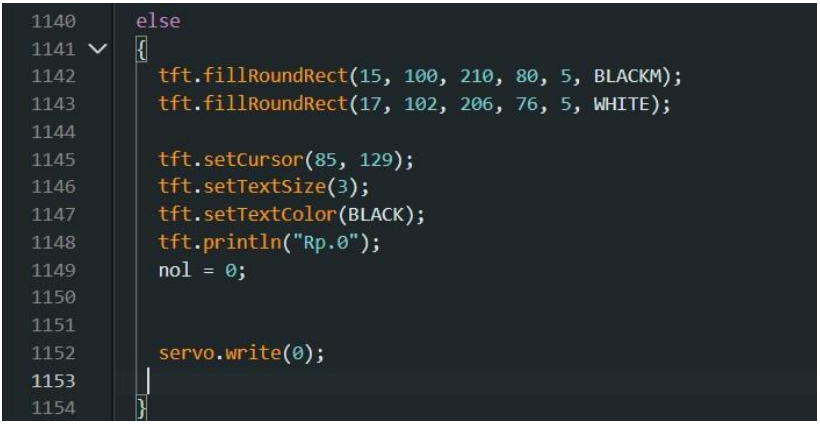
Gambar 4. 16 Source Code *Motor Servo* fungsi 5

Pada gambar diatas merupaka Source Code yang terdapat blok else if yang memeriksa kondisi lainnya. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut:

1. Kondisi else if: Ini adalah kondisi kedua yang dievaluasi jika kondisi pertama tidak terpenuhi. Dalam kondisi ini, kita memeriksa apakah nilai r1, g1, dan b1 berada dalam rentang tertentu.
2. Blok kode di dalam kondisi else if akan dieksekusi jika kondisi tersebut terpenuhi.
3. tft.fillRoundRect(): Fungsi ini mengisi suatu area persegi panjang dengan sudut melengkung pada layar TFT. Pada baris ini, kita mengisi area persegi panjang dengan sudut melengkung menggunakan warna BLACKM.

4. tft.*setCursor*(): Fungsi ini mengatur posisi kursor pada layar TFT menggunakan koordinat x dan y.
5. tft.*setTextSize*(): Fungsi ini mengatur ukuran teks pada layar TFT.
6. tft.*setTextColor*(): Fungsi ini mengatur warna teks pada layar TFT.
7. tft.println(): Fungsi ini mencetak teks pada layar TFT diikuti dengan karakter baru.
8. seratus = 100;: Nilai variabel seratus diatur menjadi 100.
9. servo_30.*write*(): Fungsi ini menggerakkan servo yang terhubung ke pin 30 ke posisi 65.
10. delay(): Fungsi ini menunda eksekusi program selama 3000 milidetik (3 detik).
11. servo_30.*write*(position2);: Setelah jeda 3 detik, servo yang terhubung ke pin 30 digerakkan ke posisi yang ditentukan oleh variabel position2.

Dengan demikian, jika kondisi pada baris kedua terpenuhi, maka blok kode di dalamnya akan di eksekusi dan *Motor Servo* akan bergerak.



```
1140    else
1141    {
1142      tft.fillRoundRect(15, 100, 210, 80, 5, BLACKM);
1143      tft.fillRoundRect(17, 102, 206, 76, 5, WHITE);
1144
1145      tft.setCursor(85, 129);
1146      tft.setTextSize(3);
1147      tft.setTextColor(BLACK);
1148      tft.println("Rp.0");
1149      nol = 0;
1150
1151
1152      servo.write(0);
1153
1154    }
```

Gambar 4. 17 *Source Code Motor Servo* fungsi 6

Pada gambar diatas merupakan *Source Code* terdapat blok else yang akan dieksekusi jika kedua kondisi sebelumnya tidak terpenuhi. Berikut adalah langkah-langkah yang diambil dalam fungsi tersebut:

1. else: Ini adalah blok yang akan dieksekusi jika kedua kondisi sebelumnya tidak terpenuhi.
2. Blok kode di dalam else akan dieksekusi jika kondisi tersebut tidak terpenuhi.
3. tft.*fillRoundRect*(): Fungsi ini mengisi suatu area persegi panjang dengan sudut melengkung pada layar TFT. Pada baris ini, kita mengisi area persegi panjang dengan sudut melengkung menggunakan warna BLACKM.
4. tft.*setCursor*(): Fungsi ini mengatur posisi kursor pada layar TFT menggunakan koordinat x dan y.
5. tft.*setTextSize*(): Fungsi ini mengatur ukuran teks pada layar TFT.
6. tft.*setTextColor*(): Fungsi ini mengatur warna teks pada layar TFT.

7. tft.println(): Fungsi ini mencetak teks pada layar TFT diikuti dengan karakter baru.
8. nol = 0;: Nilai variabel nol diatur menjadi 0.
9. servo.*write*(0);: Fungsi ini menggerakkan servo ke posisi 0.

Dengan demikian, jika kondisi pada baris ketiga terpenuhi, maka blok kode di dalamnya akan ekseskusi dan *Motor Servo* tidak akan bergerak.

4.3 Hasil Implementasi

Hasil implementasi disini merupakan hasil dari *prototype* atau perancangan Kotak Penyimpanan Uang (*MoneyBox Plus*) dengan berbagai sistem. Salah satunya sistem sortir ini didapatkan hasil yang di harapkan. *Motor Servo* mampu menggerakan kotak yang terdapat pada *MoneyBox Plus* dengan akurat dan memberikan respons sesuai kondisi yang telah di tentukan untuk pecahan uang Rp.50.000 dan Rp.100.000. Melalui penggunaan kondisional *if-else*, sistem dapat mengidentifikasi dengan tepat pecahan uang yang dimasukkan ke dalam *MoneyBox Plus*.

Selain itu, hasil uji coba juga memvalidasi kinerja layar TFT yang digunakan untuk menampilkan informasi mengenai pecahan uang yang terdeteksi. Informasi yang ditampilkan dengan jelas dan akurat, memudahkan pengguna dalam memonitor jumlah uang yang telah dimasukkan ke dalam kotak penyimpanan.

Dengan hasil uji coba yang positif ini, dapat disimpulkan bahwa implementasi sensor deteksi warna pada *MoneyBox Plus* telah berhasil dan sesuai dengan tujuan penelitian. Hasil ini memberikan bukti bahwa sistem dapat digunakan sebagai alat bantu penyimpanan uang yang efektif dan efisien

4.3.1 Penggunaan Layar LCD

Pada Kotak Penyimpanan Uang (*MoneyBox Plus*) ini, Modul TFT LCD *Display Shield* digunakan untuk menampilkan informasi yang tersedia pada sistem, dan juga sebagai kontrol utama pada sistem. Modul ini dapat dihubungkan secara langsung pada board Arduino Mega 2560.

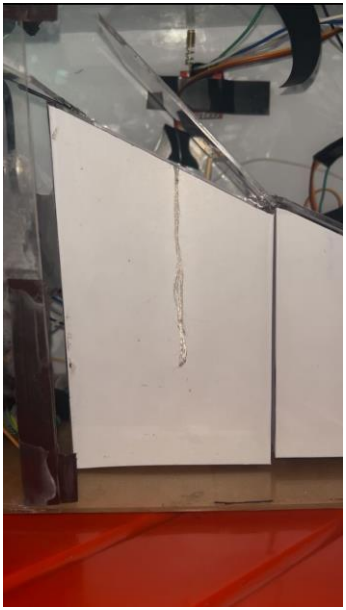
Gambar 4. 18 Penggunaan Layar LCD

4.3.2 Penggunaan *Motor Servo*

Sistem sortir pada Kotak Penyimpanan Uang (*MoneyBox Plus*) ini menggunakan *Motor Servo* yang berfungsi sebagai penggerak kotak yang terdapat didalam Kotak Penyimpanan Uang (*MoneyBox Plus*). *Motor Servo* ini dapat bergerak secara otomatis untuk dapat memisahkan uang pecahan Rp. 50.000 dan Rp. 100.000 .

Gambar 4. 19 Kotak pecahan uang RP. 100.000 terbuka

Pada gambar diatas memperlihatkan ketika uang pecahan RP. 100.000 terdeteksi maka kotak untuk uang pecahan RP. 100.000 terbuka secara otomatis.



Gambar 4. 20 Kotak pecahan uang RP. 50.000 terbuka

Pada gambar diatas memperlihatkan ketika uang pecahan RP. 50.000 terdeteksi maka kotak untuk uang pecahan RP. 50.000 terbuka secara otomatis.