Smart Pill Dispenser and App

Team #10: Yanxi Zhu, Boyuan Xie, Zijin Song

Overview

- Project objective and solution
- Hardware
- Software
- Future Work

Introduction

- A pill dispensing box designed specific for drug abuser
- A companion App for monitor patients' behavior

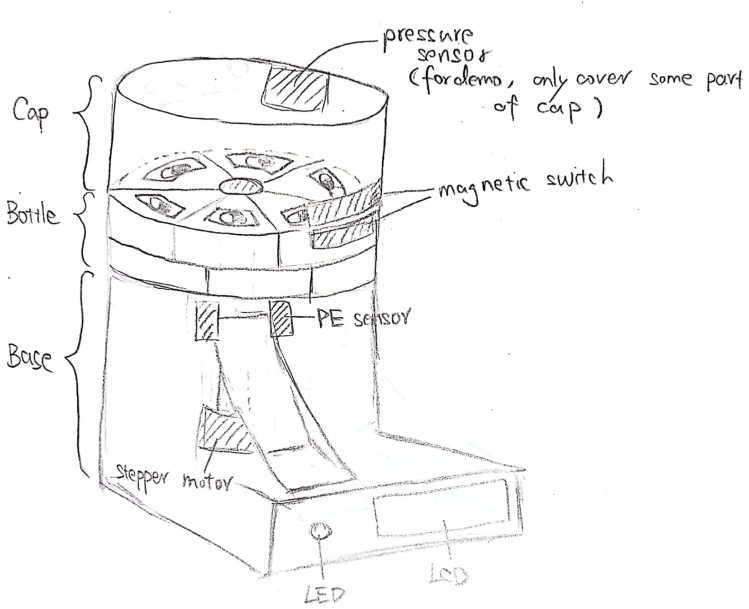
Objectives

Aim to:

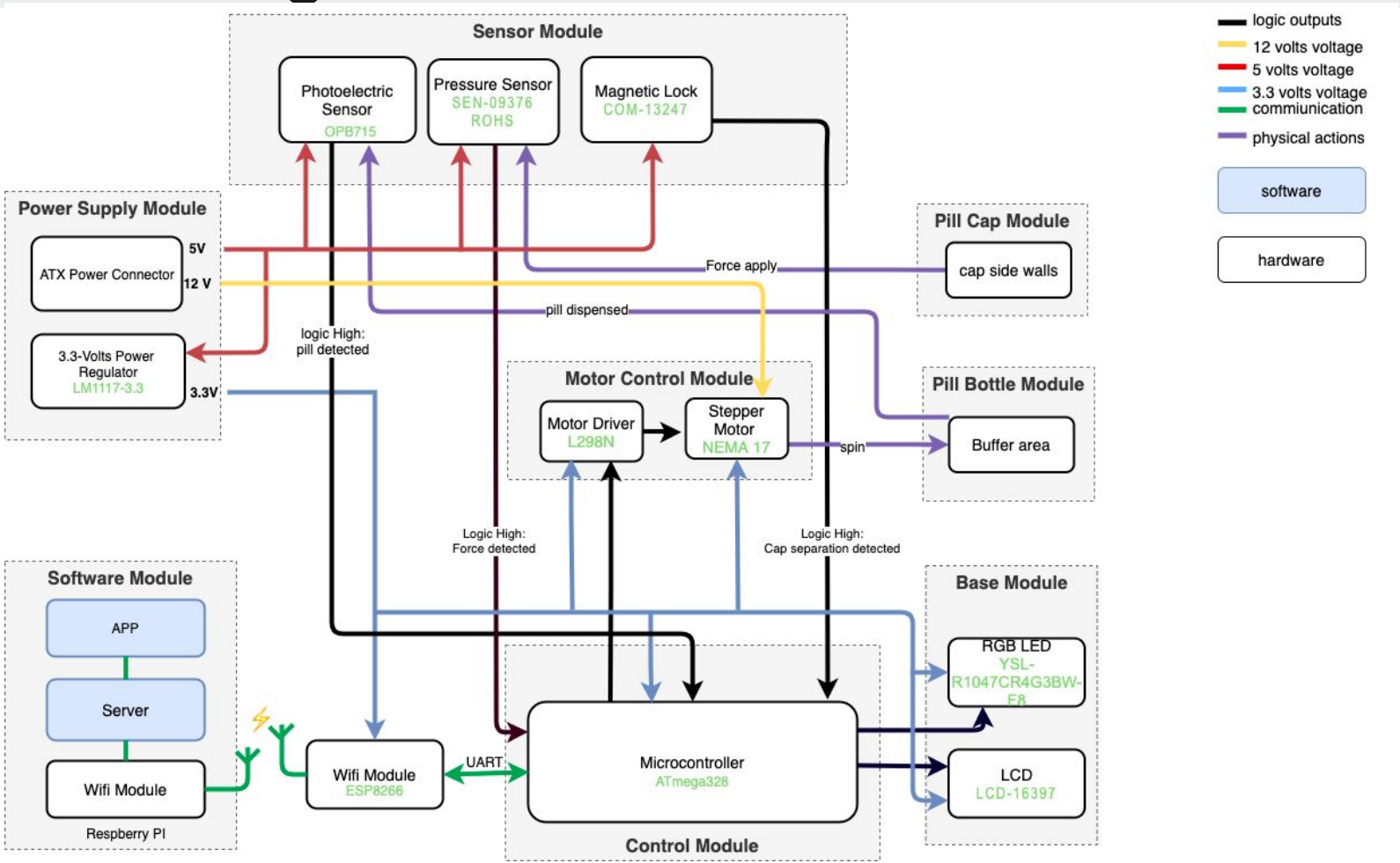
- Physically limit patients to only take allowed number of pills
- Alarm patients and doctors when violation behavior is detected

Proposed Solution

- Automatic dispenser
- Physical Prevention
- Clear Display



Block Diagram

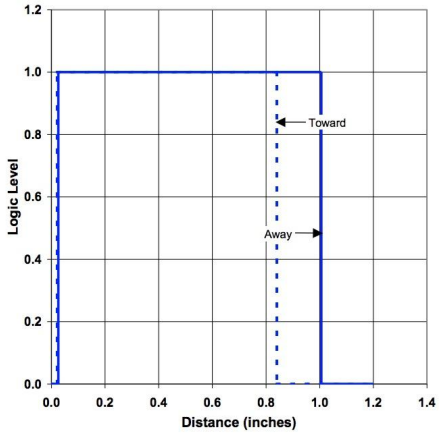


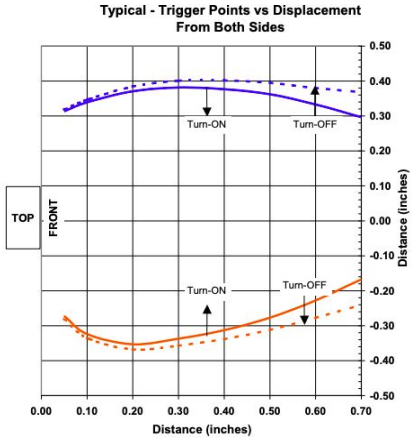
Power Supply Module

- Power Supply Module passes all tests in the lab
- Power range for 12 Volts: 12.04 - 12.4 Volts
- Power range for 5 Volts: 4.9 -5.02 Volts
- Power range for 3.3 Volts: 3.22 - 3.31 Volts

Photoelectric Sensor Module

- Giving Logic High (5 Volts) when object detected





Photoelectric Sensor

Distance from Opaque Object to Sensor	Voltage
No Object	3.5 mV
20 mm	3.7 mV
18 mm	3.5 mV
16 mm	9.0 mV
14 mm	9.0 mV
12 mm	9.0 mV
10 mm	9.0 mV
8 mm	4.38 V
6 mm	4.38 V

Table 1: Effective Distance Measurements for PE Sensor

```
void PEsensor()
{
 int PEvalue = digitalRead(peIN);
 //Serial.println(PEvalue);
 if (PEvalue == 1)
 {
  pillcount = 1;
  digitalWrite(ledG,HIGH);
 }
 else
 {
  pillcount = pillcount ;
  digitalWrite(ledG,LOW);
 }
}
```

Pressure Sensor

- Force Sensitive Resistor
- Create alarm when large forces are detected

Figure 1 - Force Curve

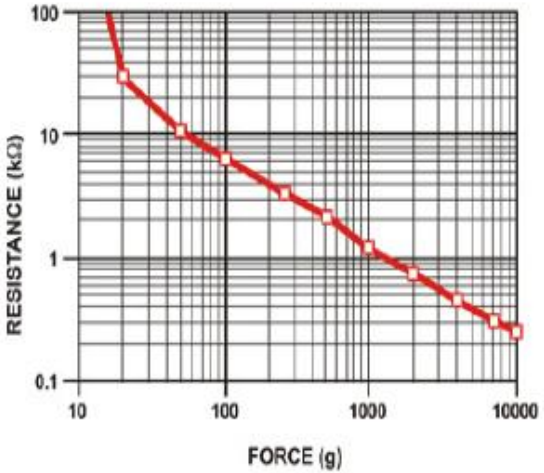
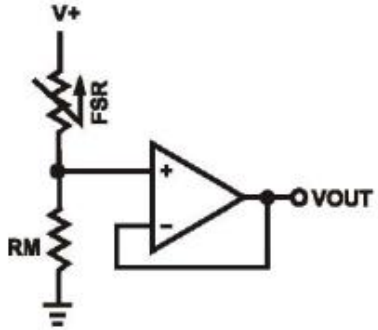
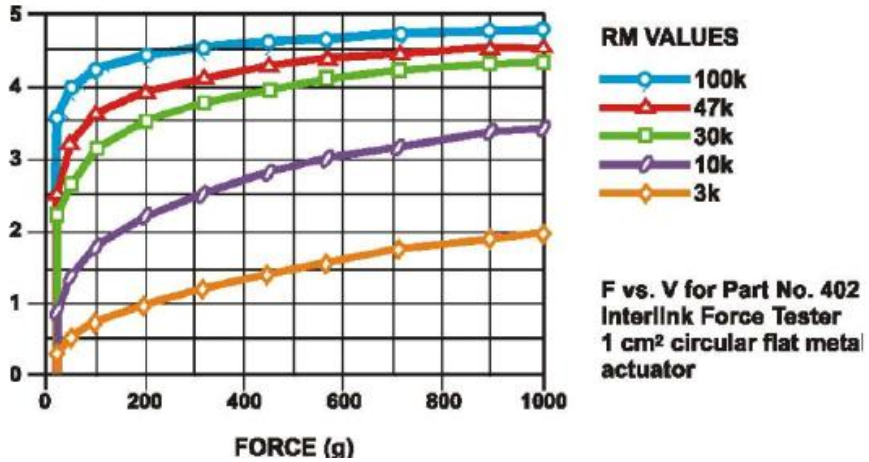


Figure 2 - Schematic



Pressure Sensor




```
void pressure()
{
  int fsrADC = analogRead(FSR_PIN);
  // If the FSR has no pressure, the resistance will be
  // near infinite. So the voltage should be near 0.
  if (fsrADC != 0) // If the analog reading is non-zero
    {
    // Use ADC reading to calculate voltage:
    float fsrV = fsrADC * VCC / 1023.0;
    // Use voltage and static resistor value to
    // calculate FSR resistance:
    // VCC is 4.98 Volts
    float fsrR = R_DIV * (VCC / fsrV - 1.0);
    //Serial.println("Resistance: " + String(fsrR) + " ohms");
    // Guesstimate force based on slopes in the FSR datasheet
    // R_DIV is 3290; 3.3 resistor is used as voltage divider
    float force;
    float fsrG = 1.0 / fsrR; // Calculate conductance
    // Break parabolic curve down into two linear slopes:
    if (fsrR <= 600)
      force = (fsrG - 0.00075) / 0.00000032639;
    else
      force =  fsrG / 0.000000642857;
    if (force > 1) {
      digitalWrite(ledR,HIGH);
      i2cSendValue(5);
      }
    else
      digitalWrite(ledR,LOW); }
    //Serial.println("Force: " + String(force) + " g");
    //Serial.println();

    else
  {
    // No pressure detected
  }
}
```

Magnetic Switch

- Provide different voltages at “on” and “off” stage

Distance between the Two Parts of the Switch	Voltage
14 mm	4.627 mV
12 mm	- 2.356 mV
10 mm	3.001 mV
8 mm	4.926 V
6 mm	4.966 V
4 mm	4.934 V
2 mm	4.933 V
0 mm	4.982 V

Table 2: Effective Distance Measurements for Magnetic Switch

```
void magneticsw()
{
int swvalue = digitalRead(swIN)
if (swvalue == 1){
   digitalWrite(ledR, LOW);
 }
 else {
 digitalWrite(ledR,HIGH);
 i2cSendValue(6);
 }
}
```

Motor - Control cascade dispensing

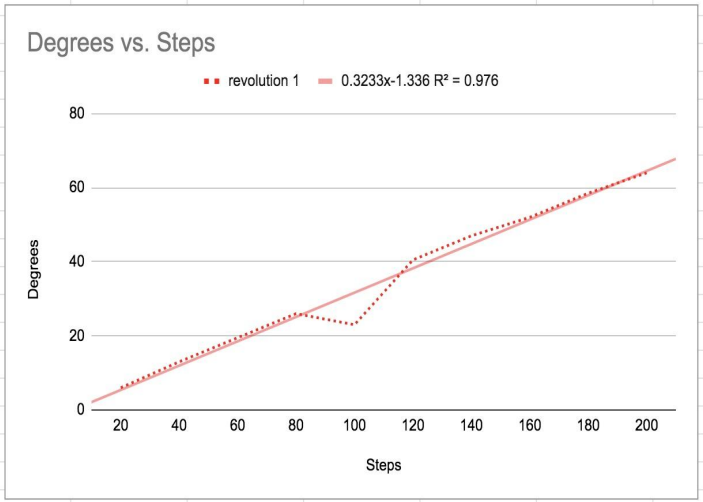
Send PWM input signal to L298N Motor driver to control NEMA 17 Stepper Motor

Carefully design motor speed and time delay to:

- Precisely load the pill to next holding position (60 degree rotation)
- Correctly count pill with PE sensor

Motor - Control cascade dispensing

Steps	revolution 1	revolution 2	revolution 3
20	6	12	19
40	13	26.5	41
60	19.5	39	60
80	26	53.5	80
100	23	67	101.5
120	40.5	82	123
140	47	93	140
160	52	104.5	157
180	58.5	117	177
200	64	128.5	194
Steps per revolution for 60 degrees			**190 forward**
			10 backward




```
void motor()
{
  stepleft = allowance - Stepcount + 1;
  if (stepleft > 0)
  {
  myStepper.step(stepsPerRevolution);
  delay(1000);

  // step one revolution in the other direction:
  //Serial.println("counterclockwise");
  myStepper.step(-10);
  delay(500);
  Stepcount = Stepcount + 1;
  myStepper.step(0);
  delay(2000);
  }
  else {
  //i2cSendValue(4);
  myStepper.step(0);
  i2cSendValue(2);
  digitalWrite(ledR,LOW);
  digitalWrite(ledY,HIGH);
  digitalWrite(ledG,LOW);

    }
}
```

LCD

- LCD should print multiple values
- Use Qwiic connectors to interface with ATmega328
- Display words and numbers on the screen

```
void i2cSendValue(int value)
{
  Wire.beginTransmission(DISPLAY_ADDRESS1); // transmit to

  Wire.write('|'); //Put LCD into setting mode
  Wire.write('-'); //Send clear display command
  if (value == 1)
  {
  Wire.print("Pills dispensing");
  }
  else if (value == 2)
  {
  Wire.print("Finish dispensing");
  }
  else if( value == 4 )
  {
  Wire.print("Numebr of Pills for this time: ");
  Wire.print(allowance);
  }
  else if( value == 5)
  {
  Wire.print("Do not break bottle!");
  }
  else if( value ==6)
  {
  Wire.print("Do not open bottle!");
  }
  else if (value == 7)
  {
  Wire.print("reset initials");
  }
  else if ( value == 8)
  {
  Wire.print("refilling");
  }
  else if (value == 9)
  {
  Wire.print("need to be refilled");
  }
  //Wire.print(value);
  else
  {
  }
  Wire.endTransmission(); //Stop I2C transmission
}
```

LCD

WIFI Module

- Esp8266
- Test:
  - Connect to local wifi
  - Transmit data in bi-direction between client and server
- Communicate between MCU and APP

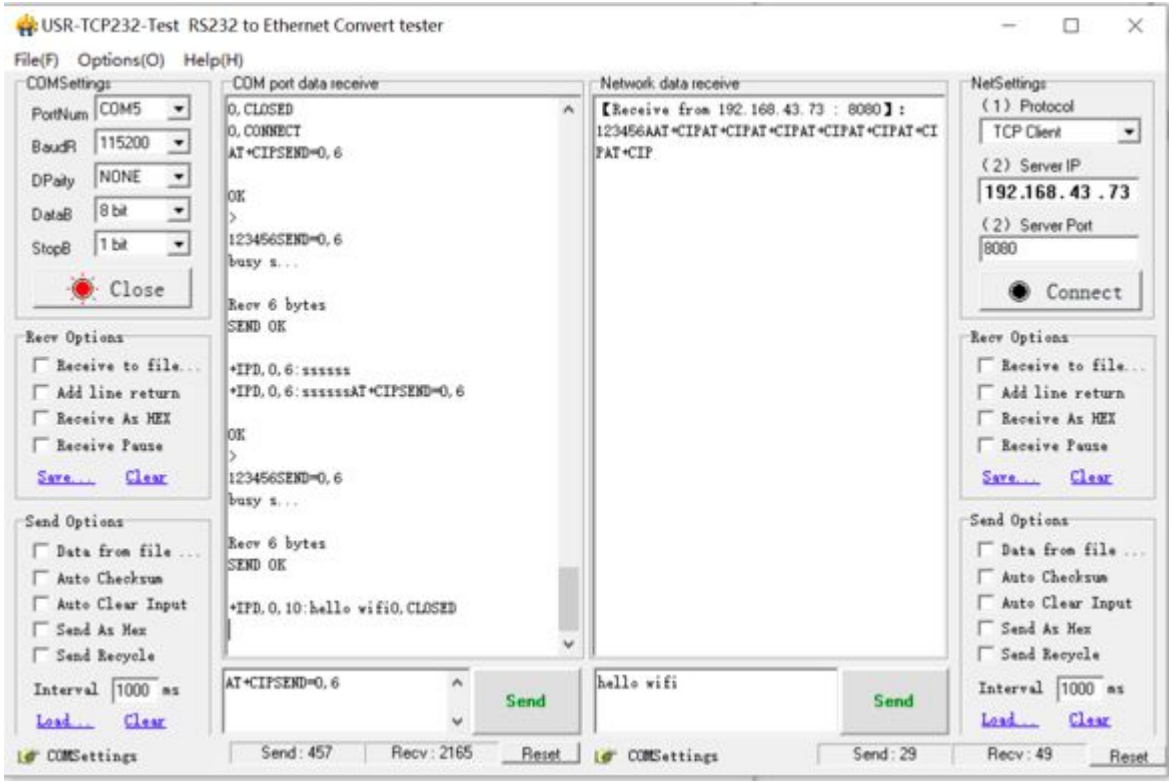
WIFI Module
- connect to wifi

Commends:

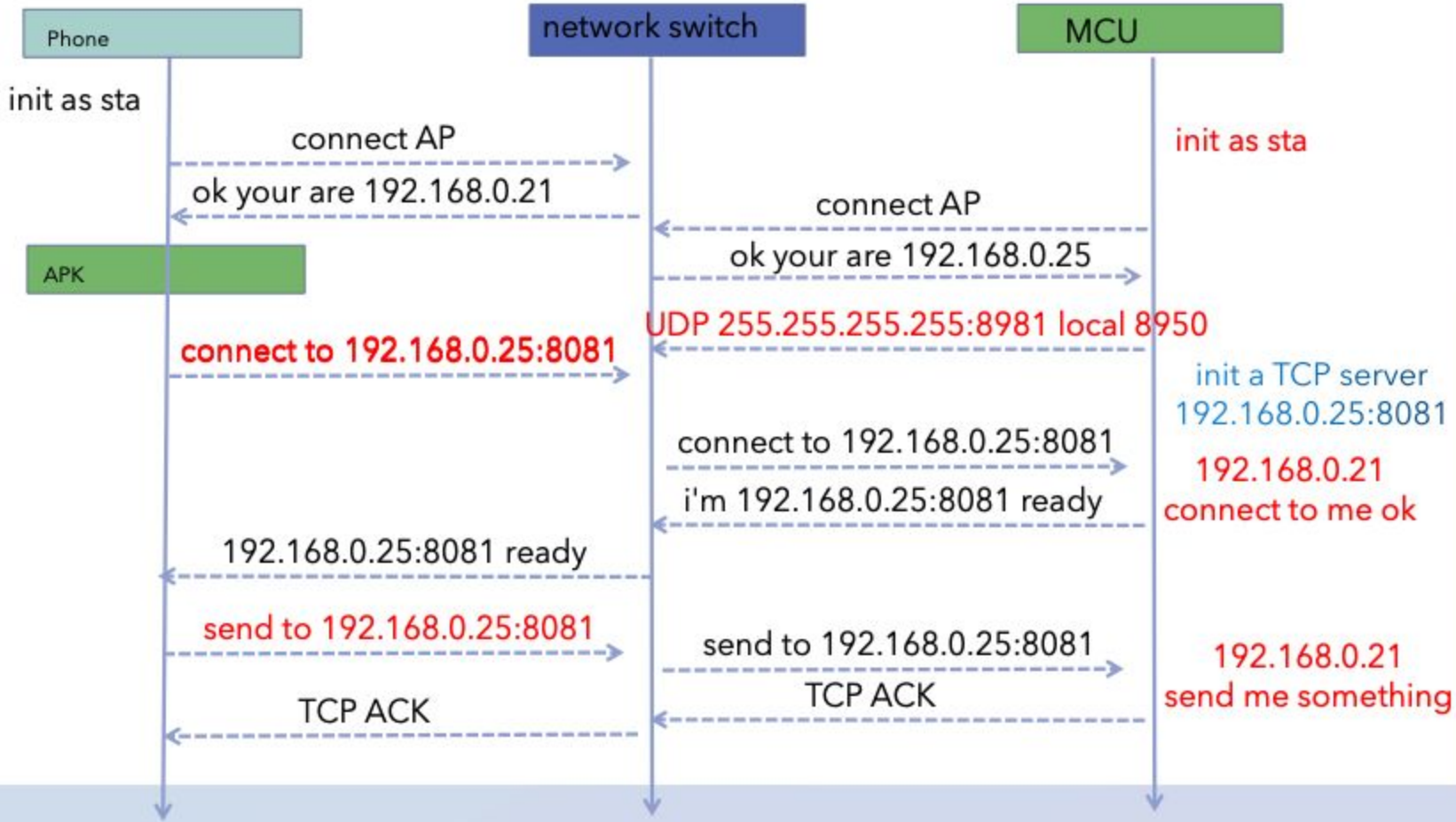
- AT+RST // reset module
- AT+CWMODE=1 // set to station mode
- AT+RST // reset module
- AT+CWLAP // list all AP
- AT+CWJAP="OPPO Reno Z","123456123456"
  // join current wifi with <ssid> and <pwd>
- AT+CWJAP? // check successfully connected to wifi

WIFI Module - transmit data

- Bi-direction



TCP client/server



Incorporate WIFI Module to the Whole System

softwareSerial_esp8266 | Arduino 1.8.14 Hourly Build 2020/09/23 10:35

softwareSerial_esp8266

```
 */
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

char send_at_cmd(char *p_at_cmd, unsigned char wait_count);
unsigned char recv_udp(char *buff, unsigned char buf_size, unsigned short wait_count);

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  Serial.println("Goodnight moon!");

  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);
}


char g_dst_ip[17];

void loop() // run over and over
{
  char buff[30];
  unsigned char len;

  send_at_cmd("AT+RST\r\n", 20);
  send_at_cmd("AT+CWMODE=1\r\n", 20);
  send_at_cmd("AT+RST\r\n", 20);
  send_at_cmd("AT+CWLAP\r\n", 20);
```

softwareSerial_esp8266 | Arduino 1.8.14 Hourly Build 2020/09/23 10:35

softwareSerial_esp8266

```
void loop() // run over and over
{
  char buff[30];
  unsigned char len;

  send_at_cmd("AT+RST\r\n", 20);
  send_at_cmd("AT+CWMODE=1\r\n", 20);
  send_at_cmd("AT+RST\r\n", 20);
  send_at_cmd("AT+CWLAP\r\n", 20);
  send_at_cmd("AT+CWJAP=\"ZWM\",\"PSS123456\"\r\n", 20);
  //send_at_cmd("AT+CWJAP=\"OPPO R15 zwm\",\"123456123456\"\r\n",  20);
  send_at_cmd("AT+CIFSR\r\n", 20);
  send_at_cmd("AT+CIPMUX=1\r\n", 20);;
  //
  send_at_cmd("AT+CIPSTART=2,\"UDP\",\"255.255.255.255\",8951,8950,0\r\n", 20); // 8950
  send_at_cmd("AT+CIPSEND=2,14\r\n", 20);
  send_at_cmd("Where are you?", 20);
  len = recv_udp(buff, 29, 1000);
  Serial.println("**parsing to get IP**\n");
  {// +IPD,2,13:192.168.50.18   parsing to get IP
    unsigned char i;
    for (i=0;i<len;i++)
    {
        if (buff[i] == ':')
          break;
    }
    memcpy(g_dst_ip, &buff[i+1], len-i-1);
    Serial.println(g_dst_ip);
  }
  Serial.println("**create server**\n");
  send_at_cmd("AT+CIPSERVER=1,8952\r\n", 20);

  delay(1000000);
}
```

Software Module

Login/Sign up

Doctor/Patient version app

Client-server communication

Login code

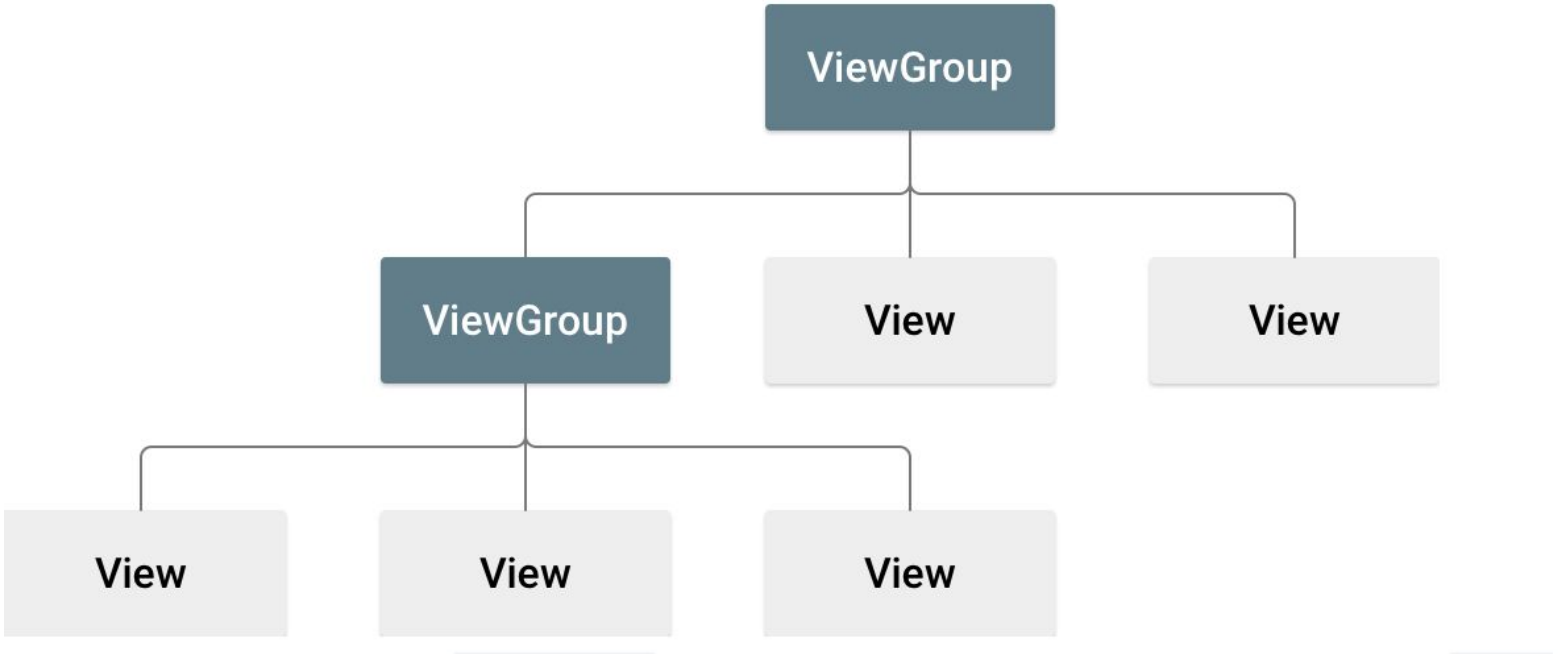
```
te static final String TAG = "LoginActivity";
te static final int REQUEST_SIGNUP = 0;

View(R.id.input_email) EditText _emailText;
View(R.id.input_password) EditText _passwordText;
View(R.id.btn_login) Button _loginButton;
View(R.id.link_signup) TextView _signupLink;

ride
c void onCreate(Bundle savedInstanceState) {
uper.onCreate(savedInstanceState);
etContentView(R.layout.activity_login);
utterKnife.bind( target: this);

loginButton.setOnClickListener((v) → { login(); });

signupLink.setOnClickListener((v) → {
        // Start the Signup activity
        Intent intent = new Intent(getApplicationContext(), SignupActivity.class);
        startActivityForResult(intent, REQUEST_SIGNUP);
        finish();
        overridePendingTransition(R.anim.push_left_in, R.anim.push_left_out);
);


c void login() {
og.d(TAG,  msg: "Login");

f (!validate()) {
   onLoginFailed();
   return;
```

```
                this.finish();
            }
        }
    }

    @Override
    public void onBackPressed() {
        // Disable going back to the MainActivity
        moveTaskToBack( nonRoot: true);
    }

    public void onLoginSuccess() {
        _loginButton.setEnabled(true);
        finish();
    }

    public void onLoginFailed() {
        Toast.makeText(getBaseContext(),  text: "Login failed", Toast.LENGTH_LONG).show();

        _loginButton.setEnabled(true);
    }

    public boolean validate() {
        boolean valid = true;

        String email = _emailText.getText().toString();
        String password = _passwordText.getText().toString();

        if (email.isEmpty() || !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            _emailText.setError("enter a valid email address");
            valid = false;
        } else {
```

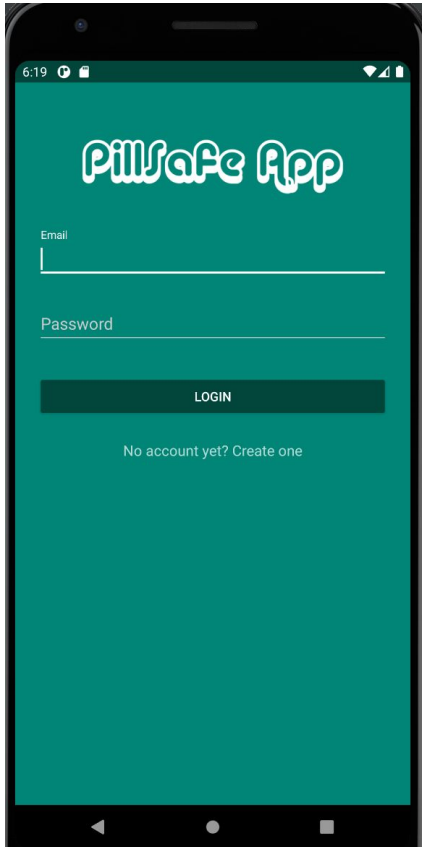
```
        _loginButton.setEnabled(false);

        final ProgressDialog progressDialog = new ProgressDialog( context: LoginActivity.this,
                R.style.AppTheme_Dark_Dialog);
        progressDialog.setIndeterminate(true);
        progressDialog.setMessage("Authenticating...");
        progressDialog.show();

        String email = _emailText.getText().toString();
        String password = _passwordText.getText().toString();

        // TODO: Implement your own authentication logic here.

        new android.os.Handler().postDelayed(
                () -> {
                        // On complete call either onLoginSuccess or onLoginFailed
                        onLoginSuccess();
                        // onLoginFailed();
                        progressDialog.dismiss();
                }, delayMillis: 3000);
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == REQUEST_SIGNUP) {
            if (resultCode == RESULT_OK) {

                // TODO: Implement successful signup logic here
                // By default we just finish the Activity and log them in automatically
                this.finish();
            }
```

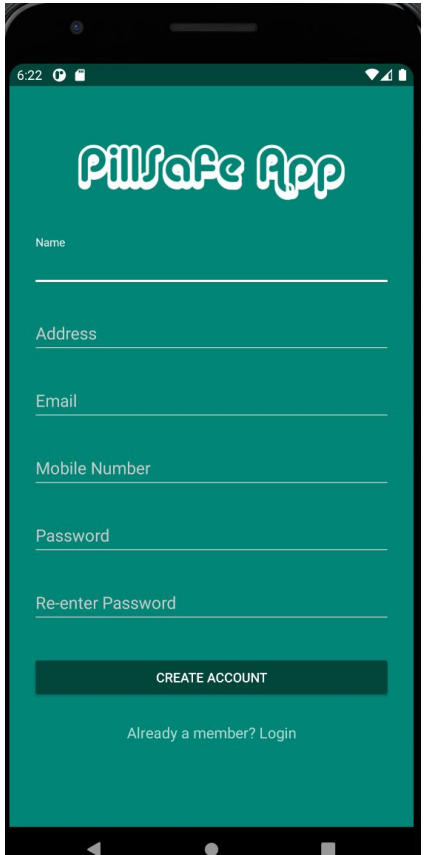
```
        } else {
            _emailText.setError(null);
        }

        if (password.isEmpty() || password.length() < 4 || password.length() > 10) {
            _passwordText.setError("between 4 and 10 alphanumeric characters");
            valid = false;
        } else {
            _passwordText.setError(null);
        }

        return valid;
    }
}
```

hierarchy of *layouts*



Sign-up/login UIs





Using socket to connect server and client

```
class ClientThread implements Runnable {

    private Socket socket;
    private BufferedReader input;

    @Override
    public void run() {

        try {
            InetAddress serverAddr = InetAddress.getByName(SERVER_IP);
            socket = new Socket(serverAddr, SERVERPORT);

            while (!Thread.currentThread().isInterrupted()) {

                this.input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                String message = input.readLine();
                if (null == message || "Disconnect".contentEquals(message)) {
                    Thread.interrupted();
                    message = "Server Disconnected.";
                    showMessage(message, Color.RED);
                    break;
                }
                showMessage( message: "Server: " + message, clientTextColor);
            }

        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }

    }
```

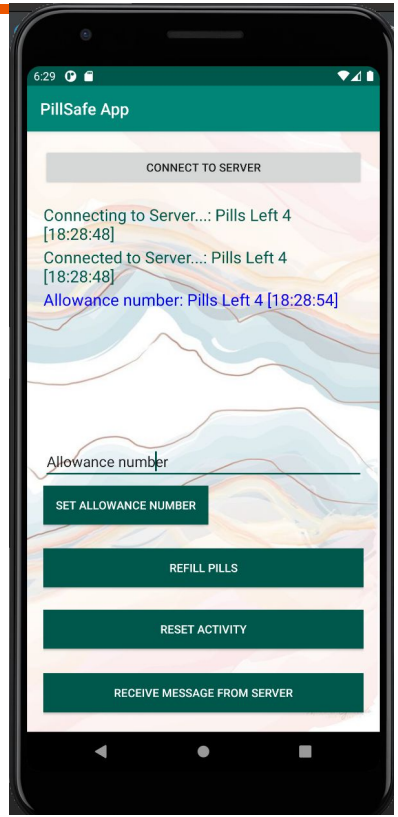
Client code

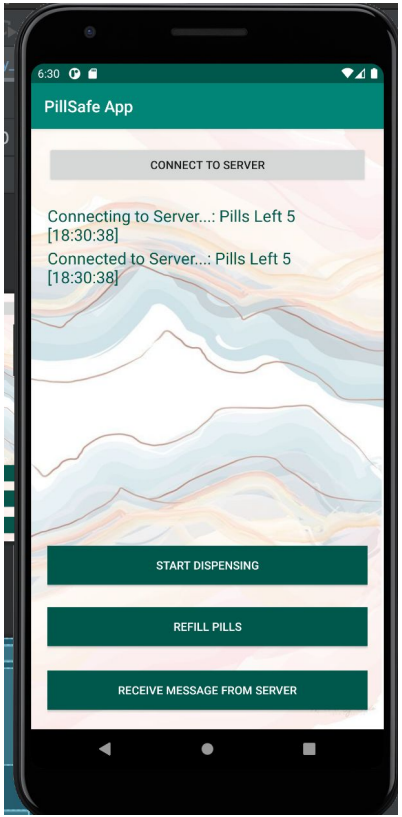
```
public class ClientActivity extends AppCompatActivity implements View.OnClickListener {

    public static final int SERVERPORT = 8080;

    public static final String SERVER_IP = "10.180.128.191";
    private ClientThread clientThread;
    public int counter = 5;
    private Thread thread;
    private LinearLayout msgList;
    private Handler handler;
    private int clientTextColor;
    private EditText edMessage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client);

        setTitle("PillSafe App");
        clientTextColor = ContextCompat.getColor( context: this, R.color.green);
        handler = new Handler();
        msgList = findViewById(R.id.msgList);
        edMessage = findViewById(R.id.edMessage);
    }

    public TextView textView(String message, int color) {
        if (null == message || message.trim().isEmpty()) {
            message = "<Empty Message>";
        }
        TextView tv = new TextView( context: this);
        tv.setTextColor(color);
        tv.setText(message + ": Pills Left " + counter + " [" + getTime() +"]");
        tv.setTextSize(20);
        tv.setPadding( left: 0,  top: 5,  right: 0,  bottom: 0);
        return tv;
    }

    public void showMessage(final String message, final int color) {
        handler.post(() -> { msgList.addView(textView(message, color)); });
    }

    @Override
    public void onClick(View view) {

        if (view.getId() == R.id.connect_server) {
            counter -= 1;
            msgList.removeAllViews();
            showMessage( message: "Connecting to Server...", clientTextColor);
            clientThread = new ClientThread();
            thread = new Thread(clientThread);
            thread.start();
            showMessage( message: "Connected to Server...", clientTextColor);
            return;
        }

        if (view.getId() == R.id.send_data) {
            String clientMessage = edMessage.getText().toString().trim();
            showMessage(clientMessage, Color.BLUE);
            if (null != clientThread) {
                clientThread.sendMessage(clientMessage);
            }
        }
    }

    class ClientThread implements Runnable {
```

Client code

```
private Socket socket;
private BufferedReader input;

@Override
public void run() {

    try {
        InetAddress serverAddr = InetAddress.getByName(SERVER_IP);
        socket = new Socket(serverAddr, SERVERPORT);

        while (!Thread.currentThread().isInterrupted()) {

            this.input = new BufferedReader(new InputStreamReader(socket.getInputS
            String message = input.readLine();
            if (null == message || "Disconnect".contentEquals(message)) {
                Thread.interrupted();
                message = "Server Disconnected.";
                showMessage(message, Color.RED);
                break;
            }
            showMessage( message: "Server: " + message, clientTextColor);
        }

    } catch (UnknownHostException e1) {
        e1.printStackTrace();
    } catch (IOException e1) {
        e1.printStackTrace();
    }

}

void sendMessage(final String message) {
    new Thread((Runnable) () -> {
        try {
            if (null != socket) {
                PrintWriter out = new PrintWriter(new BufferedWriter(
                        new OutputStreamWriter(socket.getOutputStream())),
                        autoFlush: true);
                out.println(message);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }).start();
}

}

String getTime() {
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "HH:mm:ss");
    return sdf.format(new Date());
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (null != clientThread) {
        clientThread.sendMessage("Disconnect");
        clientThread = null;
    }
}
}
```

Doctor/patient version app





Conclusion

- Successfully passed the verifications listed out for each individual module
- However, we encounter challenges when we integrated them together
  - Hardware: Failed to put components on the PCB
    - Forgot external clock and reset- button circuits when design PCB
  - Software: Failed to transfer data wirelessly
    - underestimated the difficulty of connecting server to the MCU
    - Time limit

Future Work

- Motor
  - Test speed effects to avoid issues during dispensing
- LED
  - Use one RGB LED instead of multiple LEDs
- Wifi Module
  - Wireless communication
- PCB
  - Move components to the PCB

Thank you for watching.

Feel free to ask any questions.