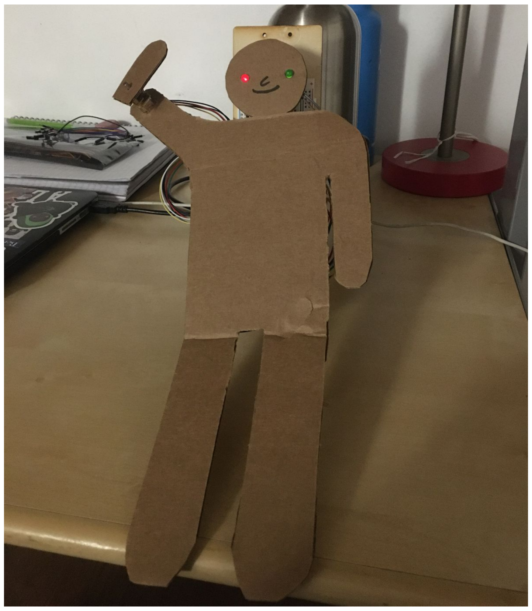
ME 102B -- Fall 2020
Final Project

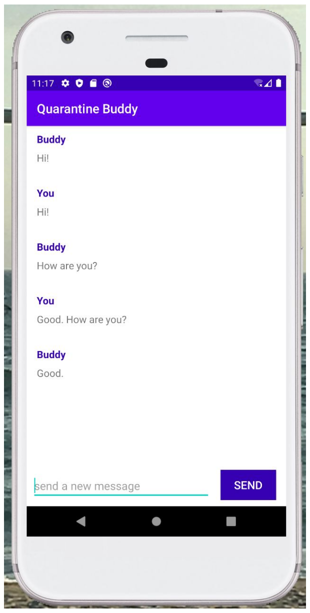
Quarantine Buddy

By Srishti Goswamy

Description of the Product:

For our final Mechatronics Design project, we were asked to create a product for our own use. While thinking about what would be most useful to me currently, I couldn't help but consider that we are currently in a global pandemic and adjusting to a new lifestyle of social distancing. I, like many others, moved back home with my family when the pandemic started, and at times want nothing more than to talk to a friend without risking either of our safety. This led me to design a Quarantine Buddy, a robot that lights up and waves at different speeds. It also connects to an Android chatbot application to help combat social distancing loneliness. From a technical standpoint, I was interested in applying mechatronics concepts from class and used the ESP32 microcontroller, potentiometer, DRV8833 dual motor driver carrier, resistors, and LEDs from the microkit. I also applied Option 1 "I'm interested in [technology X]" by including a software component in the form of an Android app. Please watch the attached video for a video demonstration of the product.




Quarantine Buddy

Buddy
Hi!

You
Hi!

Buddy
How are you?

You
Good. How are you?

Buddy
Good.

Figure 1: Quarantine Buddy's hardware (left) and software (right) components.

Electromechanical details:

<u>Interfacing between the circuit and cardboard prototype:</u> The body of the product featured a cardboard figure attached to a circuit and breadboard on a wooden backing. The breadboard is connected to the cardboard figure at two locations: the eyes and waving arm. The eyes have two holes for the LEDs to fit

snugly, and the arm has a shoulder portion that fits the DC motor body and a separate forearm portion that interfaces with the shaft. This allows the shaft to rotate freely to give the appearance of waving.

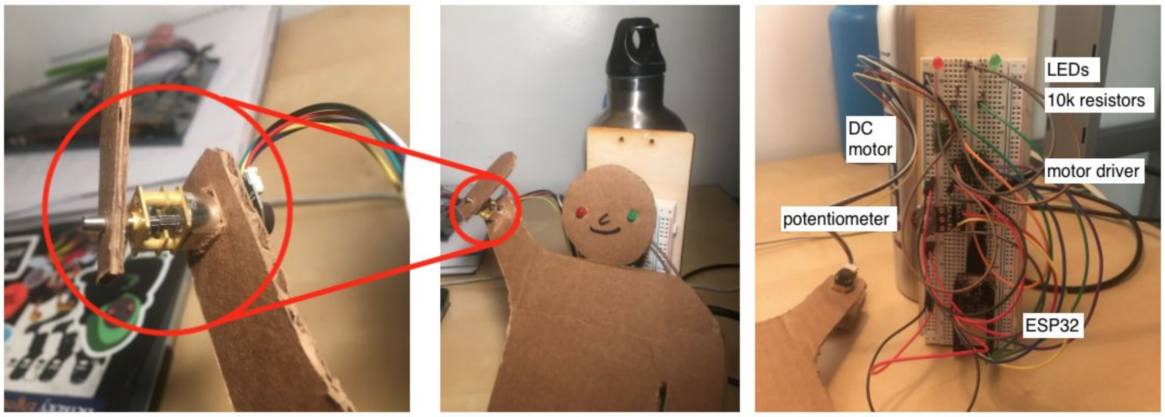


Figure 2: (Left and middle) Interface between the DC motor and cardboard figure. (Right) Circuit behind the cardboard figure, which is described in more detail in the following section.

Circuit:

The circuit involved the following parts from the microkit: an ESP32 microcontroller, linear potentiometer, DRV8833 dual motor driver carrier, Pololu DC gearmotor, two LEDs, and two 10k Ohm resistors. The LEDs only needed to turn on and off, so they were implemented using a simple circuit. The 3.3 V voltage from the ESP32 traveled from GPIO output-enabled pins 32 and 14, through the 10k Ohm resistors to prevent damage to the LEDs, and finally through the LEDs which allowed them to emit light before returning to ground. The LEDs were powered using the Arduino digitalWrite() function with a HIGH value argument.

However, the circuit and Arduino code to implement the arm waving functionality was more complex. The code utilized pulse-width modulation (PWM), a method which discretizes an electrical signal and is particularly useful for running inertial loads like motors that are unaffected by quick on-and-off switching. Using ledcSetup(), I chose the PWM channels and configured the frequency to 5000 and the resolution to 8 bits. I attached the necessary GPIO pins, A1 and 13, to their respective PWM channels using ledcAttachPin(). In the loop() function, I read the duty cycle from the potentiometer and mapped it from 4095 to 255 (2^8 - 1). Then, I used ledcWrite to set the first PWM channel to the duty cycle and the second channel to 0 for 400 milliseconds, and then set the first to 0 and the second to the duty cycle for another 400 milliseconds, in order to switch back and forth between clockwise and counterclockwise motion to create the effect of a waving arm.

In terms of the circuit, the arm waving functionality used a potentiometer that connected to the 3.3 V ESP32 voltage source, ground, and an input-enabled pin. I connected the microcontroller to Ain1 and Ain2 from the motor driver to receive duty cycle information from the potentiometer, and then connected Aout1 and Aout2 from the motor driver to the DC motor to power the motor while enabling bidirectional motion. The motor driver was also connected to ground and the 3.3 V voltage source from the microcontroller.

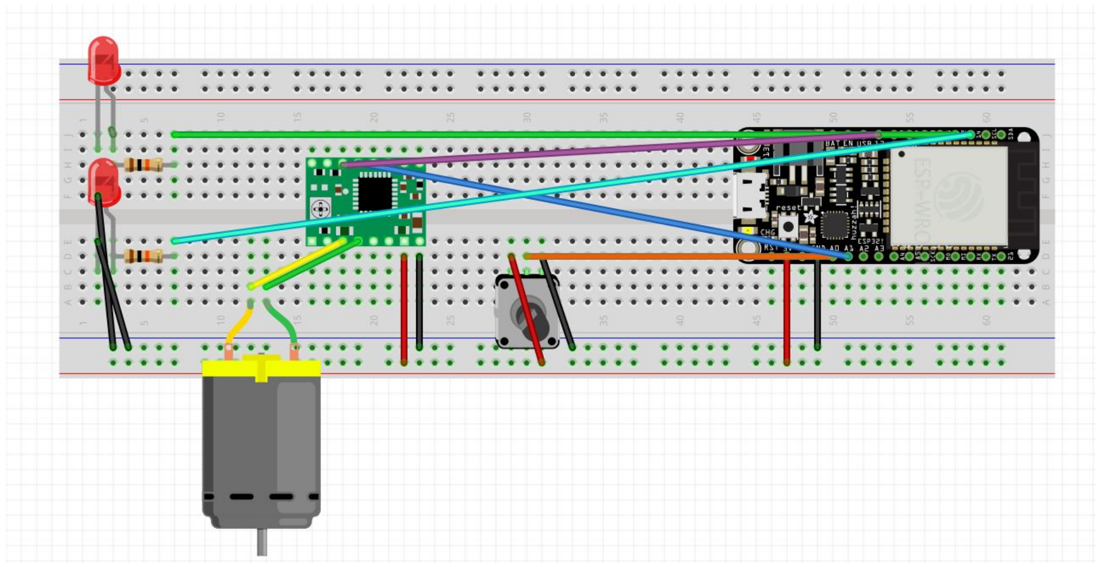
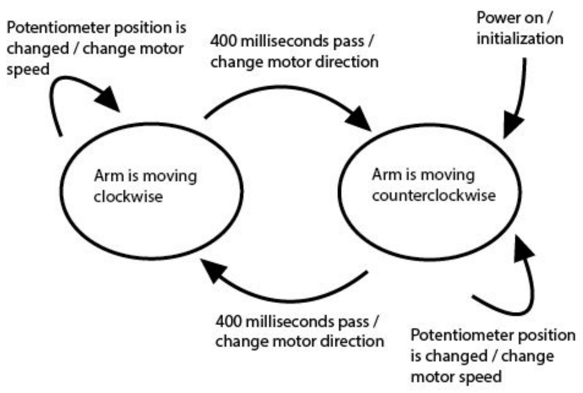


Figure 3: Circuit diagram for the product. I chose to wire the diagram exactly like the actual product for conceptual clarity, which is why some of the wires are crossed or slightly messy in the diagram.

Finite state machine:

Neither the LED functionality nor the Android applications were finite state machines, so they are omitted here, and I instead focus on the arm-waving mechanism. The FSM had two states, one where the motor moves clockwise and one where it moves counterclockwise. The Arduino code alternates between these two states every 400 milliseconds, so each 400 millisecond period that passes causes a change in motor direction and change of state. Additionally, I used a guard condition to include that changing potentiometer position changes motor speed, although it does not alter the FSM state. This data is also displayed in a bar graph, which shows the motor alternating between clockwise and counterclockwise motion every 400 milliseconds.


Potentiometer position is changed / change motor speed
400 milliseconds pass / change motor direction
Power on / initialization
Arm is moving clockwise
Arm is moving counterclockwise
400 milliseconds pass / change motor direction
Potentiometer position is changed / change motor speed

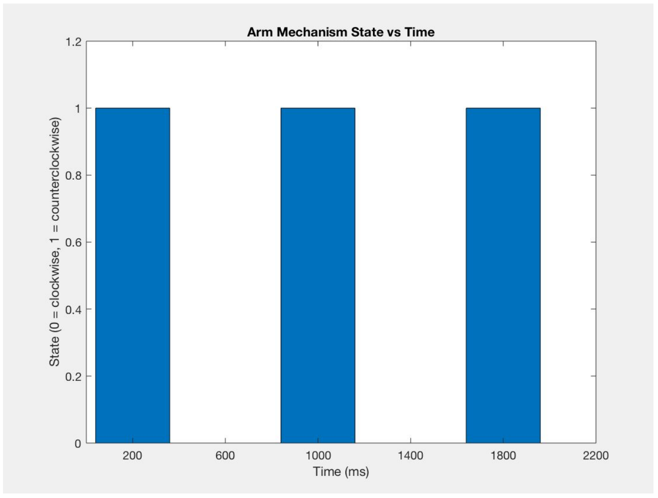


Figure 4: (Left) Finite state diagram depicting arm-waving mechanism. (Right) Graph depicting motor state vs time.

Android application:

The accompanying Android application is a simple chatbot that allows the user to send and receive messages with Quarantine Buddy. I built the application in Java using Android Studio, and it is composed of three main files: MessageActivity.java, MessageAdapter.java, and Message.java, as well as layout xml files. In the Android lifecycle, an activity corresponds to a single screen in an app, and it is where the application's functionality is implemented. By contrast, UI elements are configured in the layout xml files. The main activity for my application was MessageActivity.java, where I initialize the app in the onCreate() method, configure an onClickListener in sendButton(), post messages using postMessage(), and determine the chatbot replies in reply(). MessageActivity.java also references MessageAdapter, a class that extends RecyclerView.Adapter and binds elements from Message class objects to the appropriate UI elements.

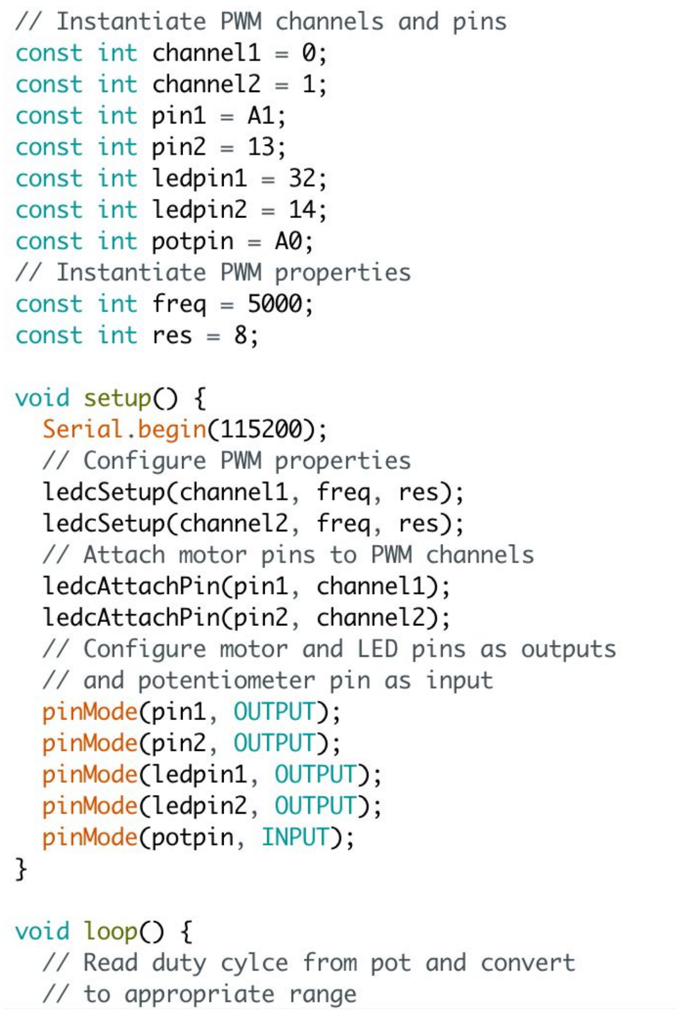
On the frontend, two message_board_layout.xml and message_cell_layout.xml determine the UI. Message_board_layout.xml utilizes a RecyclerView ViewGroup to display an unlimited number of messages. The layout for each individual message is determined within message_cell_layout.xml.

Future improvements:

Due to time and resource constraints, there were some features that I wasn't able to implement for this project but hope to implement in the future to improve the product as a whole. First, and most obviously, the cardboard figure could be improved using a 3D-printed prototype. 3D-printing has greater flexibility in terms of machining than a cardboard cut with scissors does, which would allow for a more intricate and visually pleasing design. Most 3D-printed plastics also have greater longevity than cardboard, which is especially important at the interface between the figure and the motor, which will wear down fairly quickly on the cardboard prototype.

There were also several improvements I hoped to implement in the Android application. I looked into different ways to communicate between the app and the ESP32, including using the UsbManager class to read and write to the serial monitor and hopefully turn the Quarantine Buddy on automatically when the app was launched. I began implementing this functionality in lines 44 through 55 of MessageActivity.java, but unfortunately I was not able to successfully connect to the ESP32 USB. Still, with some more time and possibly by utilizing wifi communication, this would definitely be an interesting feature to implement. Also, the current chatbot AI could be improved and expanded using natural language processing, which would use machine learning to develop authentic responses. This could be done using the Google Cloud Natural Language API.

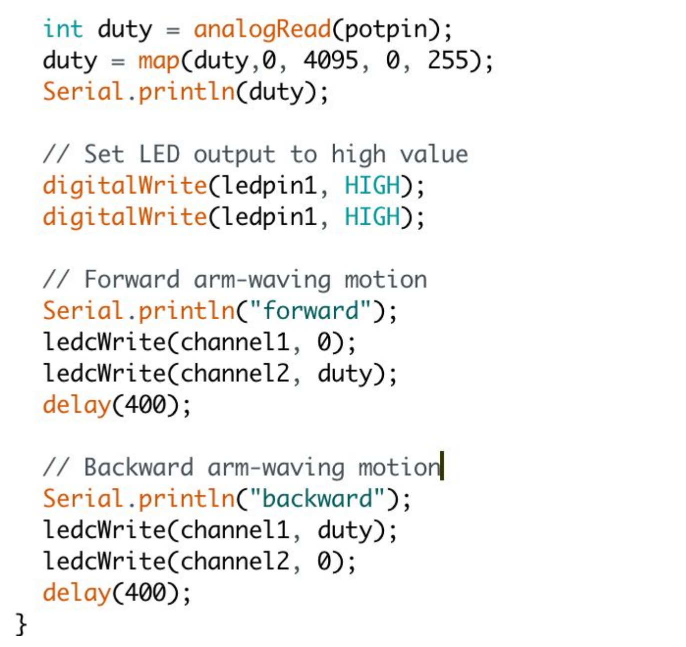
Appendix 1: Arduino Code



```
// Instantiate PWM channels and pins
const int channel1 = 0;
const int channel2 = 1;
const int pin1 = A1;
const int pin2 = 13;
const int ledpin1 = 32;
const int ledpin2 = 14;
const int potpin = A0;
// Instantiate PWM properties
const int freq = 5000;
const int res = 8;

void setup() {
  Serial.begin(115200);
  // Configure PWM properties
  ledcSetup(channel1, freq, res);
  ledcSetup(channel2, freq, res);
  // Attach motor pins to PWM channels
  ledcAttachPin(pin1, channel1);
  ledcAttachPin(pin2, channel2);
  // Configure motor and LED pins as outputs
  // and potentiometer pin as input
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(ledpin1, OUTPUT);
  pinMode(ledpin2, OUTPUT);
  pinMode(potpin, INPUT);
}

void loop() {
  // Read duty cylce from pot and convert
  // to appropriate range
```



```
  int duty = analogRead(potpin);
  duty = map(duty,0, 4095, 0, 255);
  Serial.println(duty);

  // Set LED output to high value
  digitalWrite(ledpin1, HIGH);
  digitalWrite(ledpin1, HIGH);

  // Forward arm-waving motion
  Serial.println("forward");
  ledcWrite(channel1, 0);
  ledcWrite(channel2, duty);
  delay(400);

  // Backward arm-waving motion
  Serial.println("backward");
  ledcWrite(channel1, duty);
  ledcWrite(channel2, 0);
  delay(400);
}
```

Appendix 2: Android Code

```java
package com.example.me_102b_project_app;

import android.content.Context;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.hardware.usb.UsbEndpoint;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.hardware.usb.UsbManager;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Map;

public class MessageActivity extends AppCompatActivity {

    private RecyclerView RecyclerView;
    private RecyclerView.Adapter Adapter;
    private ArrayList<Message> Messages = new ArrayList<Message>();

    EditText input;
    RelativeLayout layout;
    Button sendButton;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        byte[] DATA = new byte[] {1};
        int TIMEOUT = 100;

        /*UsbManager manager = (UsbManager) getApplicationContext().getSystemService(Context.USB_SERVICE);
        Map<String, UsbDevice> devices = manager.getDeviceList();
        UsbDevice device = devices.get("/dev/zcu.SLAB_USBtoUART");
        UsbDeviceConnection connection = manager.openDevice(device);
        System.out.println(devices);
        UsbEndpoint endpoint = device.getInterface(0).getEndpoint(0);
        connection.claimInterface(device.getInterface(0), true);
        connection.bulkTransfer(endpoint, DATA, DATA.length, TIMEOUT);*/

        setContentView(R.layout.message_board_layout);

        layout = (RelativeLayout) findViewById(R.id.message_layout);
        input = (EditText) layout.findViewById(R.id.input);
        sendButton = (Button) layout.findViewById(R.id.send_button);

        RecyclerView = (RecyclerView) findViewById(R.id.message_recycler);
        RecyclerView.setHasFixedSize(true);
        RecyclerView.setLayoutManager(new LinearLayoutManager(this));

        sendButton();
        sendFirstMessage();
        setAdapter();
    }

    private void sendFirstMessage() {
        Message newComment = new Message("Hi!", "Buddy");
        Messages.add(newComment);
    }

    private void sendButton() {
        sendButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String message = input.getText().toString();
                if (TextUtils.isEmpty(message)) {
                    input.requestFocus();
                } else {
                    postMessage(message, "You");
                    reply(message);
                    input.setText("");
                }
```

```java
            }
        });
    }

    private void setAdapter() {
        Adapter = new MessageAdapter(this, Messages);
        RecyclerView.setAdapter(Adapter);
        RecyclerView.smoothScrollToPosition(Messages.size
() - 1);
    }

    private void postMessage(String messageText, String
name) {
        Message newMessage = new Message(messageText, name
);
        Messages.add(newMessage);
        setAdapter();
    }

    private void reply(String message) {
        if (message.equals("Hi!")) {
            postMessage("How are you?", "Buddy");
        } else if (message.equals("Good. How are you?")) {
            postMessage("Good.", "Buddy");
        } else {
            postMessage("Sorry, I didn't understand that."
, "Buddy");
        }
    }
}


```

```java
package com.example.me_102b_project_app;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

public class MessageAdapter extends RecyclerView.Adapter {

    private Context Context;
    private ArrayList<Message> Messages;

    public MessageAdapter(Context context, ArrayList<
Message> messages) {
        Context = context;
        Messages = messages;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(
ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(Context).inflate(R.
layout.message_cell_layout, parent, false);
        return new MessageViewHolder(view);
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder
holder, int position) {
        Message message = Messages.get(position);
        ((MessageViewHolder) holder).bind(message);
    }

    @Override
    public int getItemCount() {
        return Messages.size();
    }
}

class MessageViewHolder extends RecyclerView.ViewHolder {

    public RelativeLayout MessageCellLayout;
    public TextView UsernameTextView;
    public TextView MessageTextView;

    public MessageViewHolder(View itemView) {
        super(itemView);
        MessageCellLayout = itemView.findViewById(R.id.message_cell_layout);
        UsernameTextView = MessageCellLayout.findViewById(R.id.username_text_view);
        MessageTextView = MessageCellLayout.findViewById(R.id.message_text_view);
    }

    void bind(Message message) {
        UsernameTextView.setText(message.name);
        MessageTextView.setText(message.text);
    }
}


```

```java
package com.example.me_102b_project_app;

public class Message {

    public String text;
    public String name;

    Message(String text, String name) {
        this.text = text;
        this.name = name;
    }
}


```

```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/
apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/message_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/message_recycler"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_above="@+id/send"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:transcriptMode="alwaysScroll"
        app:stackFromEnd="true" />

    <RelativeLayout
        android:id="@+id/send"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="#ffffff"
        android:paddingBottom="10dp"
        android:paddingLeft="0dp"
        android:paddingRight="0dp"
        android:paddingTop="5dp">

        <EditText
            android:id="@+id/input"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/send_button"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="16dp"
            android:layout_toLeftOf="@+id/send_button"
            android:gravity="top"
            android:hint="send a new message"
            android:inputType="textShortMessage" />

        <Button
            android:id="@+id/send_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginRight="16dp"
            android:background="@color/colorPrimaryDark"
            android:text="send"
            android:textColor="#FFFFFF"
            android:textSize="18sp" />
    </RelativeLayout>

</RelativeLayout>

```

```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/message_cell_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:orientation="vertical"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="10dp">

    <LinearLayout
        android:id="@+id/ll"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/username_text_view"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="username"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="16dp"
            android:textStyle="bold" />
    </LinearLayout>

    <TextView
        android:id="@+id/message_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ll"
        android:paddingBottom="8dp"
        android:paddingTop="8dp"
        android:text="asdfghjk"
        android:textSize="16sp" />

</RelativeLayout>
```