Morse Code Lab

In this lab you will be creating an application that will translate sentences into Morse Code. The application should allow you to type a message into your application and then blink the Morse Code translation using the LED on the Arduino.

Learning Objectives:

- Learn how to use a USB accessory with the Android Tablet
- Create a functional dual component application
- Get experience with Android and Arduino programming

Before You Start:

Download blink_app.ino, BlinkApp.zip, morse_code_shell.ino and MorseCodeShell.zip.

Part 1 (Blink App):

The code for the Blink App has already been written for you. It is a simple app with a button that will toggle an LED on the Arduino on and off. Installing and running this app will help you to understand how to create new Android accessories using the Arduino and complete the rest of this lab.

Arduino:

To start, connect the Arduino to a USB port on your computer using the USB A to B cable. Check to make sure the green "ON" LED is lit.

Open blink_app.ino with the Arduino IDE. You will be prompted to move the file into a folder called "blink_app", press OK to continue.

Notice the setup() and loop() functions. The setup function is called once when the Arduino is powered on, or when the the reset button is pressed. After that, the loop function will be called continuously.

To compile and upload code to the Arduino, you first must select the correct board and serial port.
Board selection can be found under ***Tools*** > ***Board*** > ***Arduino Mega 2560 or Mega ADK***.

To determine the correct serial port, open the ***Control Panel*** on your computer and select ***Hardware and Sound*** > ***Devices and Printers***. On this screen, you should see a device labeled *“Arduino Mega ADK”* followed by the correct COM port. Finally, in the Arduino IDE, select ***Tools*** > ***Serial Port*** > ***COM?*** (where ? is number you just located).

Finally, click the “Upload” button in the top left corner of the Arduino IDE.

Android:

In Android Studio, import the the BlinkApp project for Android by going to ***File*** > ***Import Project…*** and following the dialog to complete the import.

After importing the project, review the code in Blinker.java and try to understand it. Pay close attention to the blinkLED() method as you will be implementing something similar in the next part of the lab.

Connect your Android device to the computer, and then install the app on your device.

Finally, disconnect your Android device from the computer and connect it to the USB port on the Arduino. This should launch the BlinkApp on your device. Press the button to toggle LED on the Arduino board.

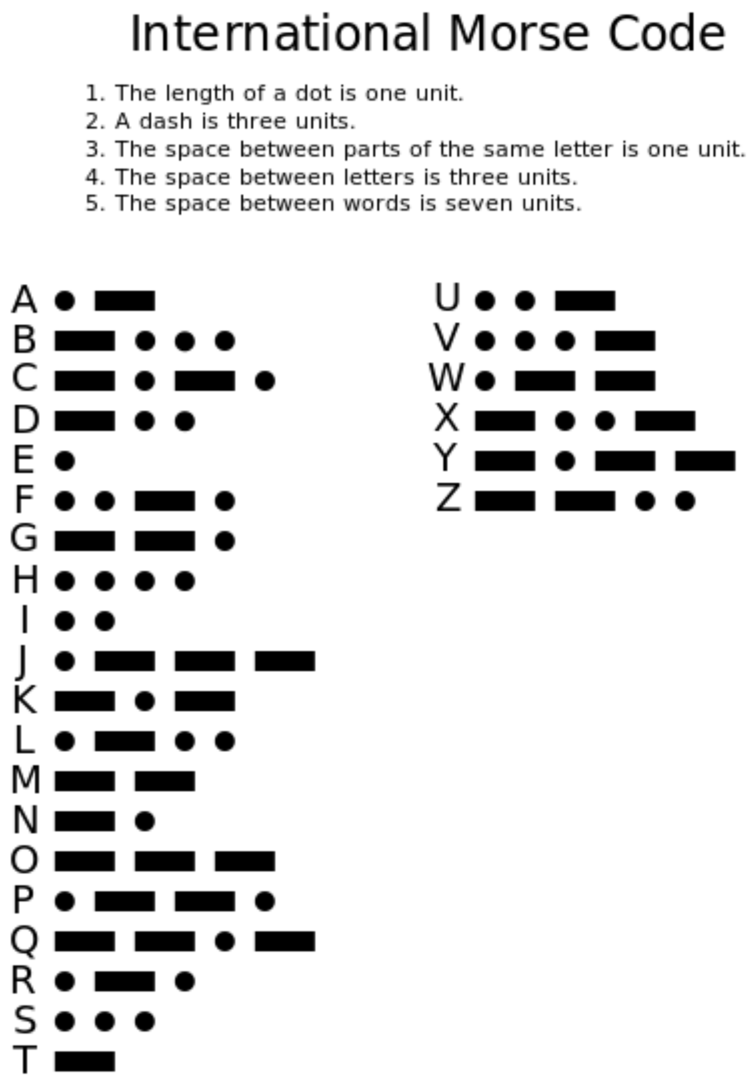
Demo this to your TA.

Part 2 (Morse Code Generator):

For this part of the lab, you will be creating an app which allows you to enter a message on your Android device and display it in Morse Code on the Arduino by toggling the same LED as the BlinkApp.

What is Morse Code?

A message in Morse Code consists of a series of dashes and dots (or longs and shorts). These dashes and dots can then be displayed by turning the LED on for the correct number of time units *(longs and shorts)* and then off for the correct number of units between parts of a letter, whole letters, and words. The rules for displaying morse code can be found in the figure below. *(Note that for this lab the code to convert* *between text and morse code has already been written for you. Your task will be to interpret the morse code and display it correctly).*



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬		
L	● ▬ ● ●		
M	▬ ▬		
N	▬ ●		
O	▬ ▬ ▬		
P	● ▬ ▬ ●		
Q	▬ ▬ ● ▬		
R	● ▬ ●		
S	● ● ●		
T	▬		

Arduino:

Open morse_code_shell.ino with the Arduino IDE like in the previous part. You should notice a very similar layout to blink_app.ino.

For the Arduino portion of this lab, you will be required to complete the *displayMorseCode()* function, to toggle the LED on and off to display the message in morse code. The *displayMorseCode()* method accepts the message and the length as

parameters. We need to interpret the message toggle the LED on and off to display the morse code.
The message consists of the following values, which have been defined as constants:

- SHORT: *a dot (or short) in morse code.*
- LONG: *a dash (or long) in morse code.*
- LETTER: *the end of a letter in morse code.*
- WORD: *the end of a word in morse code.*
- STOP: *the end of the morse code message.*

For example, the message “SOS” would be encoded as (SHORT, SHORT, SHORT, LETTER, LONG, LONG, LONG, LETTER, SHORT, SHORT, SHORT, LETTER, WORD, STOP).

Implement this method by turning the LED on and off the same way it is done in blink_app.ino. In order to toggle the LED for a certain length of time, use the *delay()* function. In your code, call *delay(UNIT)* to pause execution of your code for UNIT milliseconds. Using the UNIT constant will allow the length of one unit to easily be changed without altering your code.

You can now compile and upload your code to the Arduino.

Android:

In Android Studio, import the the MorseCodeGeneratorShell project for Android by going to ***File*** > ***Import Project…*** and following the dialog to complete the import.

Now that you have imported the project, you will need to add a EditText and Button to the application’s layout. Pressing the button should call the *blinkLED()* method in your java code.

The EditText should be configured so that it only accepts letters and spaces. It will also not accept more than 25 characters of input. This can be done by using the ***android:digits*** and ***android:maxLength*** attributes in your EditText in the layout.

Writing openAccessory() Method:

The openAccessory method is used to create the connection to the USB accessory, there are a few instance variables available to use.

1. To start we need to open the connection to do this add the following line *mFileDescriptor = mUsbManager.openAccessory(accessory);*

mFileDescriptor is a file descriptor to handle the accessory. mUsbManager contains a method to detect if there is an accessory connected to the tablet and attempts to connect to the device.

2. Below this line we need to make sure that the file descriptor actually contains the opened accessory so we can use it. To do this we want write an if statement to check that mFileDescriptor is not null. If it is not null, then the accessory was opened successfully. If it is null, then the accessory failed to open and we should log this by adding the line: *Log.d(TAG, “accessory open fail”);* This will display in Android Studio that the accessory failed to open.

3. If the descriptor isn’t null, we need to save the accessory and open an input and output stream. To do this, add the following lines inside your if statement.

   ```
   mAccessory = accessory;
   FileDescriptor fd = mFileDescriptor.getFileDescriptor();
   mInputStream = new FileInputStream(fd);
   mOutputStream = new FileOutputStream(fd);
   Log.d(TAG, “accessory opened”);
   ```

Writing blinkLED() Method:

The *blinkLED()* method gets the text from your EditText, encodes it, and send its to the Arduino to be displayed.

The message can be encoded by using the *encodeMessage()* method. This will convert the message to the morse code format and return a byte array containing the encoded message.

```
private byte[] encodeMessage(String msg);
```

Look at the BlinkApp example above for ideas on how to send the encoded message to the Arduino. Finally, upload the app to your Android device, and then connect it to the Arduino board as before.

Demo the lab to your TA.