This write up will describe one way to communicate between the Arduino and a smart phone, over BlueTooth. A product called RoboRemo (Robotic Remote), which allows a programmer to set up an app that transmits strings over BlueTooth, Wifi or USB.

The app shown in the image below was set up, using four sliders. One (yellow) moves a stepper motor, while the others control the Red, Green and Blue intensity on a tri color LED. A demo of how these are setup and how it is used will be done in class. Also refer to the manual at http://www.roboremo.com/downloads.html

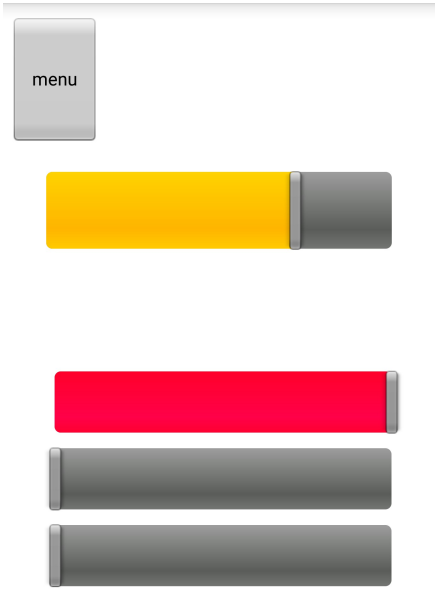


Figure RoboRemo-1. My First RoboRemo Setup.

Once a slider is placed, on the RoboRemo display, it has a variety of options that need to be set. These are shown in Figure RoboRemo-2. The first to be set is the ID, which is the characters that will be sent when that slider is changed. Then the "set min,max" option should be set, defining the range of numbers the slider will return. Also important is the "send when moved" versus "send when released" option. Of course other things will also play into the effectiveness of the interface, such as the color of the slider, but that will be something that will need to be experimented with.

Now the yellow-motor slider has an id of M and a number range of 0 to 100. It is also set to "send when released". We want it to only be sent on the release of the slider since we don't want to have the motor starting and stopping constantly when we touch this part of the screen. For example, if this slider were moved to the center of its range and released, it would transmit the string "M 50\n". Note the last character is commonly called the terminator, and in this case it is the new line character, which is a common string terminator. Thus the Arduino must read these strings in and respond accordingly.

The sliders that control the red, green and blue LED's are set with id's of R, G, and B respectively. Also their range is 0 to 255, the same as the analogWrite function. Finally they are set to "send on move", since this won't be a problem with the LED intensities and it allows the user to adjust the intensity interactively.

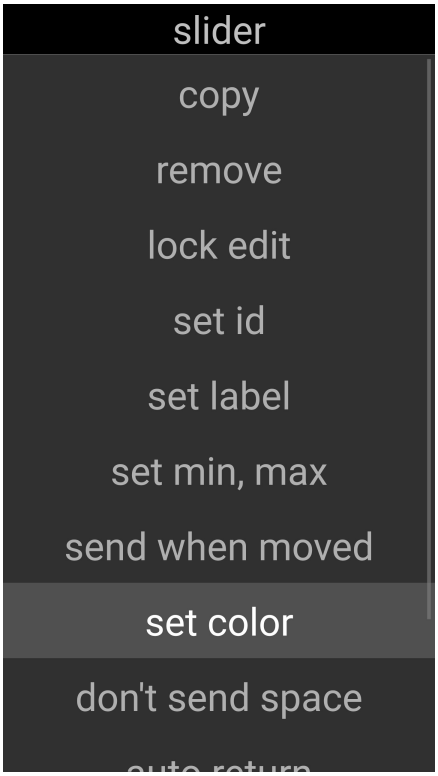


Figure RoboRemo-2. Options for a slider.

Other types of controls (commonly called widgets) are available as shown in Figure RoboRemo-3.

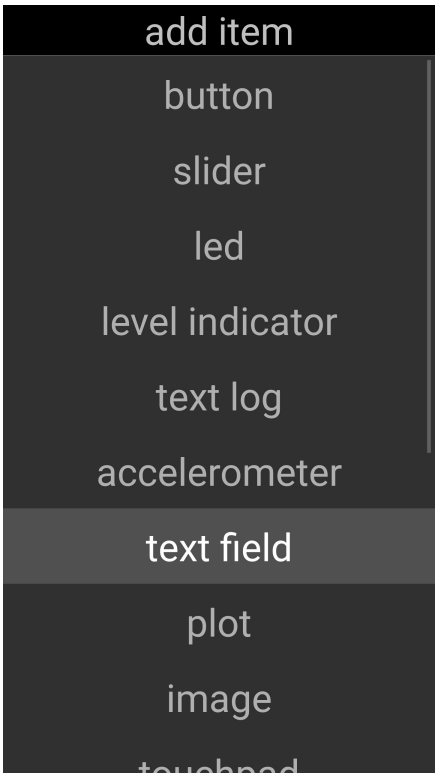


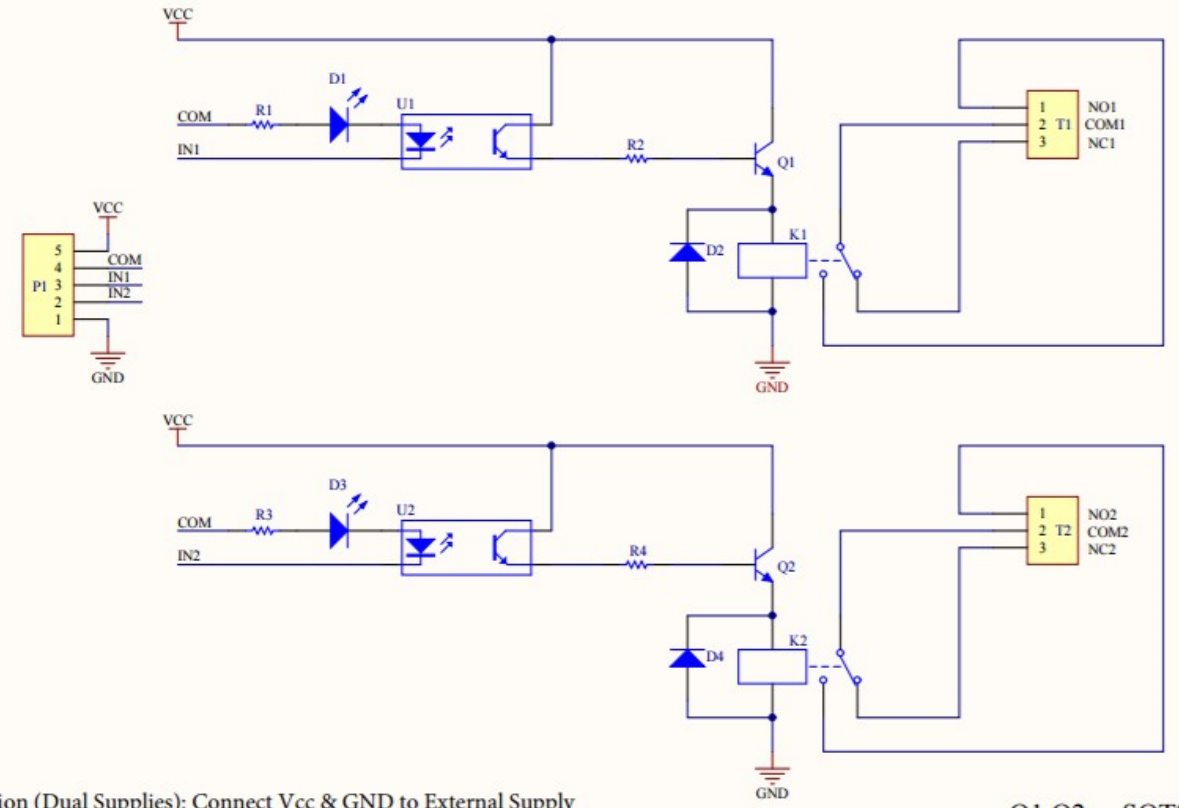
Figure RoboRemo-3. Other Widgets.

Appendix A: Code for Controlling Motor and Colored LED

```
#include <Stepper.h>

// Setup stepper motor controller.
Stepper StepperOne(100, A0, A1, A2, A3);

// Stepper Motor motion parameters
int StepperLocation = 0;    // Current location Arbitrary
int StepperDestination = 0; // Desired Destination
int StepInput, StepperMoving = 0; // Variables for input and feedback.
int Reset = 0;

// function to handle moving stepper forward.
void StepperMove()
{
	// if destination is lower than location
	if (StepperLocation > StepperDestination)
	{
		if (digitalRead(12) == 0 // if limit indicator is active
			&& !Reset)           // but not trying to correct.
		{
			// Set Destination as two steps back.
			StepperDestination = StepperLocation + 3;
			Reset = 1; // Set flag to allow motion when sensor low
		}
		else // Take a negative step.
		{
			StepperOne.step(-1);
			StepperLocation--;
		}
	}
	else if (StepperLocation < StepperDestination) // if opposite
	{
		if (digitalRead(12) == 0  // if limit indicator is active
			&& !Reset)            // but not trying to correct.
		{
			// Set Destination as two steps back.
			StepperDestination = StepperLocation - 3;
			Reset = 1;    // Set flag to allow motion when sensor low          }
		else // Take a positive step.
		{
			StepperOne.step(1);
			StepperLocation++;
		}
	}
	else // if we are at our destination
	{
		if (StepperMoving)
		{
			Serial.println("At Destination");
			StepperMoving = 0;
			Reset = 0;
		} // End of feedback if

	} // End of Pos. Neg. at destination if
} // End of StepperMove
```

```
#define STEP_INTERVAL 20
unsigned long StepTimer;
unsigned long Timer;
unsigned long CurrentStepInterval = 200; // Initial step timer

// Simple timer to flash LED.
#define LED_INTERVAL 500
unsigned long LedTimer;

#define RedLed   11
#define GreenLed 10
#define BlueLed   9

// Character string and pointer for incoming characters.
char IncomingChar[128];
int  IncomingPointer = 0;

// Run once, To set up system
void setup()
{
	// Sets time of individual steps.
	StepperOne.setSpeed(30);

	// Initialize LED timer and pin
	LedTimer = millis();
	pinMode(13, OUTPUT);

	analogWrite(RedLed, 100);
	analogWrite(GreenLed, 0);
	analogWrite(BlueLed, 0);

	// Serial Com.
	Serial.begin(9600);

	// Move stepper until sensor low
	while (digitalRead(12))
	{
		StepperOne.step(-1);
	} // Move to sensor low position

	// Move back until sensor high
	while (!digitalRead(12))
	{
		StepperOne.step(1);
	} // Move to zero position

} // End of setup

// Code that is run continuously.
void loop()
{
	char Incoming;
	int IncomingValue;

	// LED Flashing Timer.
	if (millis() - LedTimer >= LED_INTERVAL)
	{
		// Toggle LED.
		if (digitalRead(13))
			digitalWrite(13, LOW);
		else
			digitalWrite(13, HIGH);
		LedTimer += LED_INTERVAL; // Update Timer.

	} // End of Led Timer.
```

```
	if (millis() - StepTimer >= CurrentStepInterval)
	{
		StepTimer = millis(); // Update timer.
		// Yes it is different than we usually do.
		// But due to the varying time interval,
		// this works more consistently.

		StepperMove(); // Update stepper.

		// Adjust Stepper time interval, ramping up ,then down
		if (abs(StepperLocation - StepperDestination) > 10)
		{
			if (CurrentStepInterval > 20) // Provided we are not moving to fast.
			{
				// Speed up as we get to moving.
				CurrentStepInterval -= 20;
			}
		}
		else // We are getting close to our destination
		{
			// Make interval larger until it is 200 milliseconds.
			if (CurrentStepInterval < 200)
				CurrentStepInterval += 20;
		}
	} // End of Stepper Timer if.

	// Check for incoming serial data.
	if (Serial.available())
	{
		Incoming = Serial.read();

		// Decode incoming serial data.
		if (Incoming != '\n') // If not at terminator
		{
			// place character in buffer.
			IncomingChar[IncomingPointer++] = Incoming;
			// Move pointer ahead one.
		}
		else // With terminator in.
		{
			IncomingChar[IncomingPointer] = 0; // and terminator to string.
			// Decode Results
			switch (IncomingChar[0])
			{
			case 'M': // Set motor position (M #\n)
				StepperDestination = atoi(&IncomingChar[1]);
				break;
			case 'R': // Set Red Led  (R #\n)
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(RedLed, IncomingValue);
				break;
			case 'G': // Set Green Led  (G #\n)
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(GreenLed, IncomingValue);
				break;
			case 'B': // Set Blue Led  (B #\n)
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(BlueLed, IncomingValue);
			} // End of decode switch

			IncomingPointer = 0; // Reset pointer

		} // End of Terminator check
	} // End of Serial available if
} // End of loop
```



Operational Notes:

1: For Complete Isolation Operation (Dual Supplies): Connect Vcc & GND to External Supply COM to Microcontroller 5V, IN1 & IN2 to Controller Output Port(s)

2: For Signal Isolation Only (Single Supply): Connect COM to Controller Power Supply, IN1 & IN2 to Controller Output Port(s)

Q1,Q2:	SOT95P240-3N Transistor
D1,D2:	SMD LED
R1,R3:	1K Ohm SMD Resistor
R2,R4:	1.0 Ohm SMD Resistor
U1,U2:	FL817C OptoCoupler
K1,K2:	SRD-05DC-SL-C Relay

Title: Dual 5V Opto-Isolated Relay Module		
Size: A4	Number: 32707-MP	Revision: 3/16
Date: 2016/1/8		Sheet of
File:		Drawn By:

Code for Relay Control

```
// put your setup code here, to run once:
void setup()
{
	// Set pins as output and deactivate.
	pinMode(A0, OUTPUT);
	digitalWrite(A0, HIGH);
	pinMode(A1, OUTPUT);
	digitalWrite(A1, HIGH);

	// Set up serial port.
	Serial.begin(9600);

} // End of setup

// put your main code here, to run repeatedly:
void loop()
{
	// Check for incoming data
	if (Serial.available())
	{
		// Read data and take appropriate action.
		switch (Serial.read())
		{
		case '1': // If a 1 received, activate relay.
			digitalWrite(A0, LOW);
			break;
		case '0': // If a 0 received, deactivate.
			digitalWrite(A0, HIGH);
			break;
		default: // otherwise do nothing.
			break;
		}
	}
} // End of loop.
```