Projekt: Buzz Lightning

Idee

Die Idee für unser Projekt ist eine Lichtsteuerung, mit der man die Intensität mehrerer Lichter mithilfe eines Distanzsensors regeln kann.

Dabei soll zwischen zwei Modis unterschieden werden:

- Steuerung der Intensität mit absoluten Werten
- Steuerung der Intensität relativ zu aktuellem Wert

Außerdem wollen wir mithilfe eines Drehpotentiometer zwischen mehreren Lichtern auswählen können.

Als Szenario haben wir eine reale Situation gewählt: Die Beleuchtung einer Wohnung mit mehreren Lichter. Jedoch haben wir dieses Szenario etwas verändert, damit die Lichtsteurung besser wahrnehmbar ist:

- Überflüssige Texturen werden weggelassen
- Es gibt keine Lichteinflüsse von außen
- Die Lichter sind in einem rötlichen Farbton

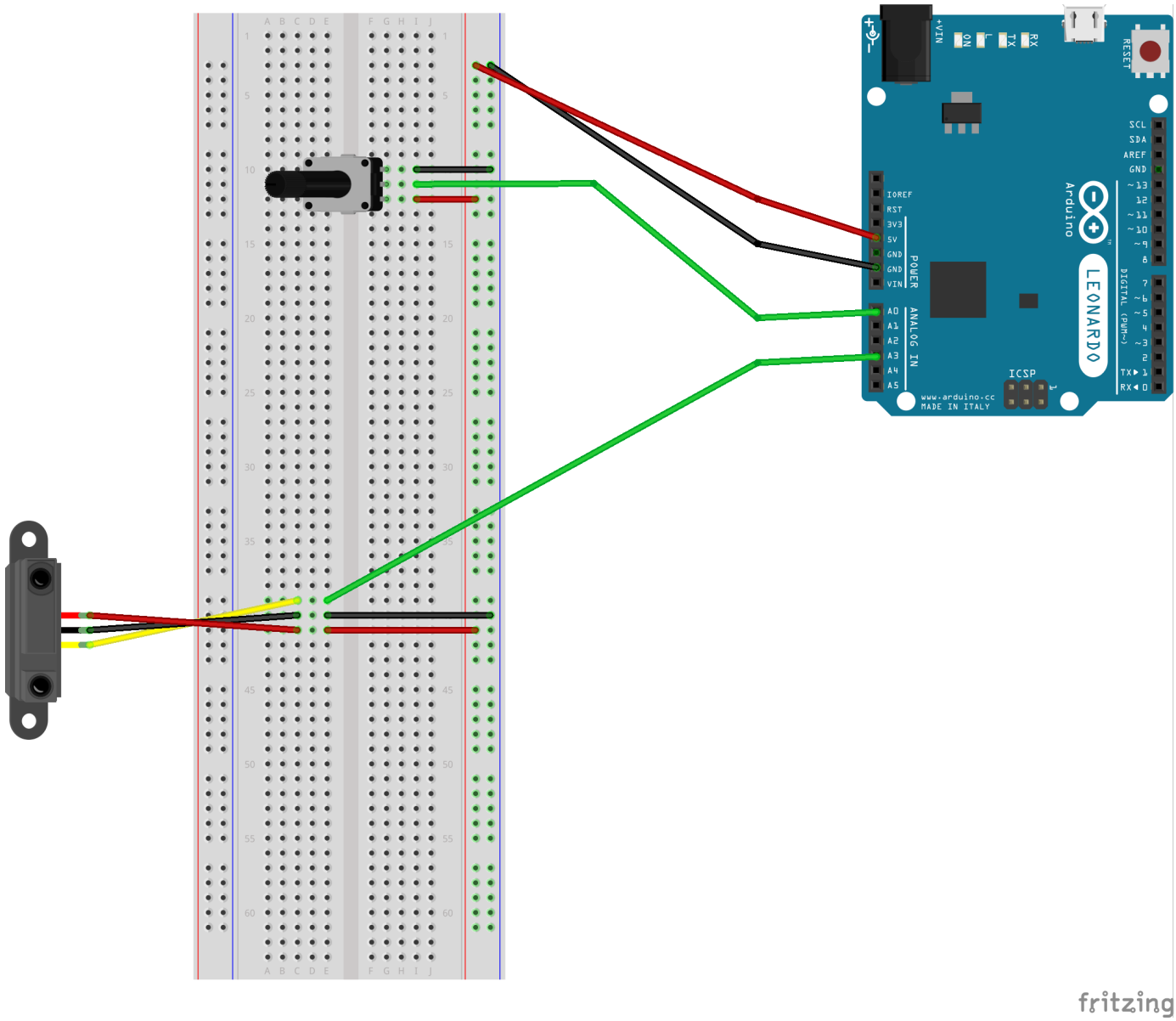
Kriterien

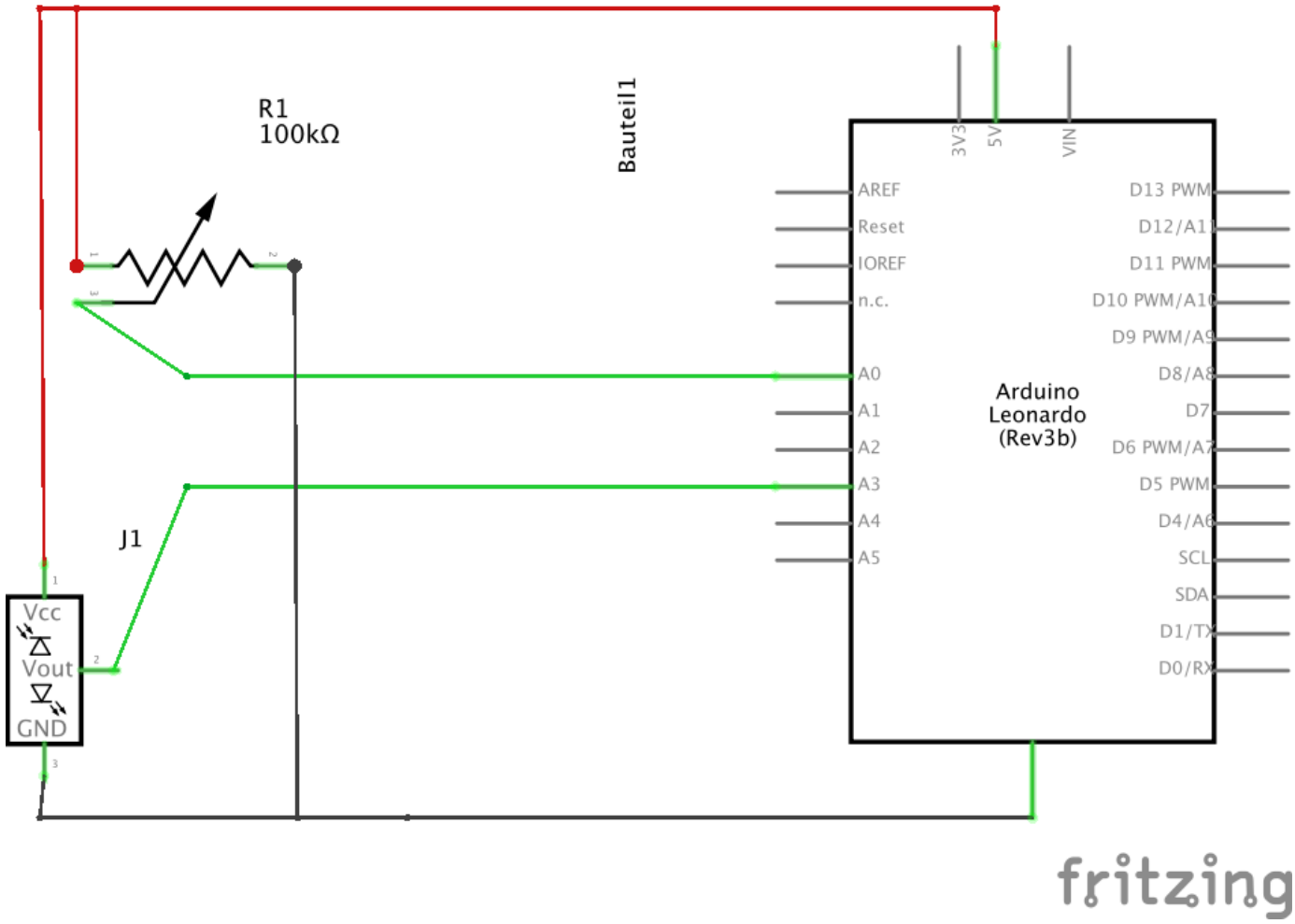
TODO

Umsetzung

Controlleraufbau

Zur Eingabe werden ein Distanzsensor und ein Drehpotentiometer verwendet. Diese werden wie in folgenden Grafiken an einen Arduiono Leonardo angeschlossen, der die Steuerung übernimmt.


LEONARDO
Arduino
POWER
ANALOG IN
DIGITAL (PWM~)
ICSP
www.arduino.cc
MADE IN ITALY
fritzing



Software

Arduino

Der Arduino regelt die Eingabewerte und übermittelt diese über den Seriellen Ausgang an unsere Software.

Dabei werden zuerst noch die Werte geglättet. Außerdem werden die Werte immer nur dann übermittelt, wenn sie einen bestimmten, vorher festgelegten Grenzwert überschreiten.

```
//Todo Some Code...
```

Außerdem war es uns wichtig, dass wir erkennen können ob die Eingabe gewollt oder ungewollt erfolgt ist.

Um die Änderungen zu unterscheiden können, haben wir uns für ein Lock-Out System entschieden. Der AnwenderInnen können die Regelung beenden, sobald sie kurz in einer Position verharren. Das System sperrt dann diese Position für einige Sekunden bevor weitere Änderungen übernommen werden können.

```
// Code
```

Java Serial2UDP

Aufgrund von Problemen mit Mono und der Seriellen Schnittstelle unter OSX haben wir ein Java Programm verwendet, dass die Seriellen Schnittstellen überwacht und sämtlichen Werte über UDP an unsere Unitysoftware weiterleitet.

Unity

Unser Unity Programm besteht aus zwei Tweilen: Einem UDP-Empfänger und einem Lichtcontroller.

UDP Empfänger

Der UDP Empfänger hört auf den UDP-Eingabeport und leitet die empfangenen Daten an den Lichtcontroller weiter.

```
private void ReceiveData() {
    client = new UdpClient(port);
    while (true)
    {

        try
        {
            // Bytes empfangen.
            IPEndPoint anyIP = new IPEndPoint(IPAddress.Any, 0);
            byte[] data = client.Receive(ref anyIP);

            // Bytes mit der UTF8-Kodierung in das Textformat kodieren.
            string text = Encoding.UTF8.GetString(data);

            String distValue = lastReceivedUDPPacket.Split('p')[0];
            int serial1 = Convert.ToInt32(distValue.Substring(1));

            String potValue = lastReceivedUDPPacket.Split('p')[1];
            int serial2 = Convert.ToInt32(potValue);

            String tag = lightController.serialToTag(serial2);
            float intensity = lightController.serialToIntensity(serial1);
            lightController.setLightIntensity(tag, intensity);
        }
        catch (Exception err)
        {
            print(err.ToString());
        }
    }
}
```

Light Controller

Der Light Controller kennt alle Lichter und wandelt die Seriellen Werte in verarbeitbaren Daten um.

Das bedeutet er übernimmt die Steuerung des Lichtes in dem er die Werte des Potentiometers umrechnet und ein Licht anwählt.

Die Lichtintensität wird ebenfalls anhand von Eingabedaten berechnet. Dabei wird der Sensorwert des Distanzsensors in Lichtintensität umgerechnet. Folgende Formel wird hierfür verwendet:

```
Intensity = Sensor_Value * (INTENSITY_MAX / SENSOR_MAX)
```

Für die Steuerung stehen daher vier Funktionen zur Verfügung:

- **serialToTag** - rechnet den Sensorwert in einen Lichttag um für die Auswahl der Lichter
- **serialToIntensity** - rechnet den Sensorwert in eine Intensität um
- **setLightIntensity** - setzt die Intensität der Lichter mit dem Tag
- **setLightIntensityRelative** - setzt die Intensität der Lichter relativ

```
/**
 * Returns the Tag for the light by the serial port mapping
 */
public String serialToTag(int serial) {
    int index = 0;

    index += Math.Max(0,  Math.Min (light_map.Count - 1, serial * light_map.Count / SE
RIAL_MAX));

    // Get the tag
    String tag = tag_list [index];

    // Highlight the selected light
    changeLight (tag);

    // Return the tag
    return tag;
}

/**
 * Returns the intensity for the serial value
 */
public float serialToIntensity(int serial) {
    serial = Math.Max (SERIAL_MIN, Math.Min (serial, SERIAL_MAX));

    float intensity = serial * (INTENSITY_MAX / SERIAL_MAX);

    return Math.Max (INTENSITY_MIN, Math.Min (intensity, INTENSITY_MAX));
}

/**
 * Set the light intensity
 */
public void setLightIntensity(String light_tag, float intensity) {
    Change change = new Change ();
    change.Tag = light_tag;
    change.Intensity = intensity;

    changes.Enqueue (change);
}

/*
 * Set the light intensity relative
 */
public void setLightIntensityRelative(String light_tag, float delta_intensity) {
    Change change = new Change ();
    change.Tag = light_tag;
    change.Intensity = light_map^light_tag][0].intensity + delta_intensity;

    changes.Enqueue (change);
}
```

Wie oben zu sehen ist, werden die Änderungen nicht direkt übernommen sondern in eine Changes-Queue gespeichert. Diese Queue wird dann innerhalb der Update Funktion abgeabeitet.

```
public void Update() {
    // If there are changes to do, do it..
    if (changes.Count > 0)
    {
        Change change = changes.Dequeue();

        // Change the light intensities
        foreach(Light light in light_map[change.Tag] ) {
            light.intensity = Math.Max(INTENSITY_MIN, Math.Min(INTENSITY_MAX, change.I
ntensity));
        }
    }
}
```

3D-Modelle

Ergebnis