Flambe Controller Spec

Revision 40

Table of Contents

1. Specification

1.1 Connection Overview

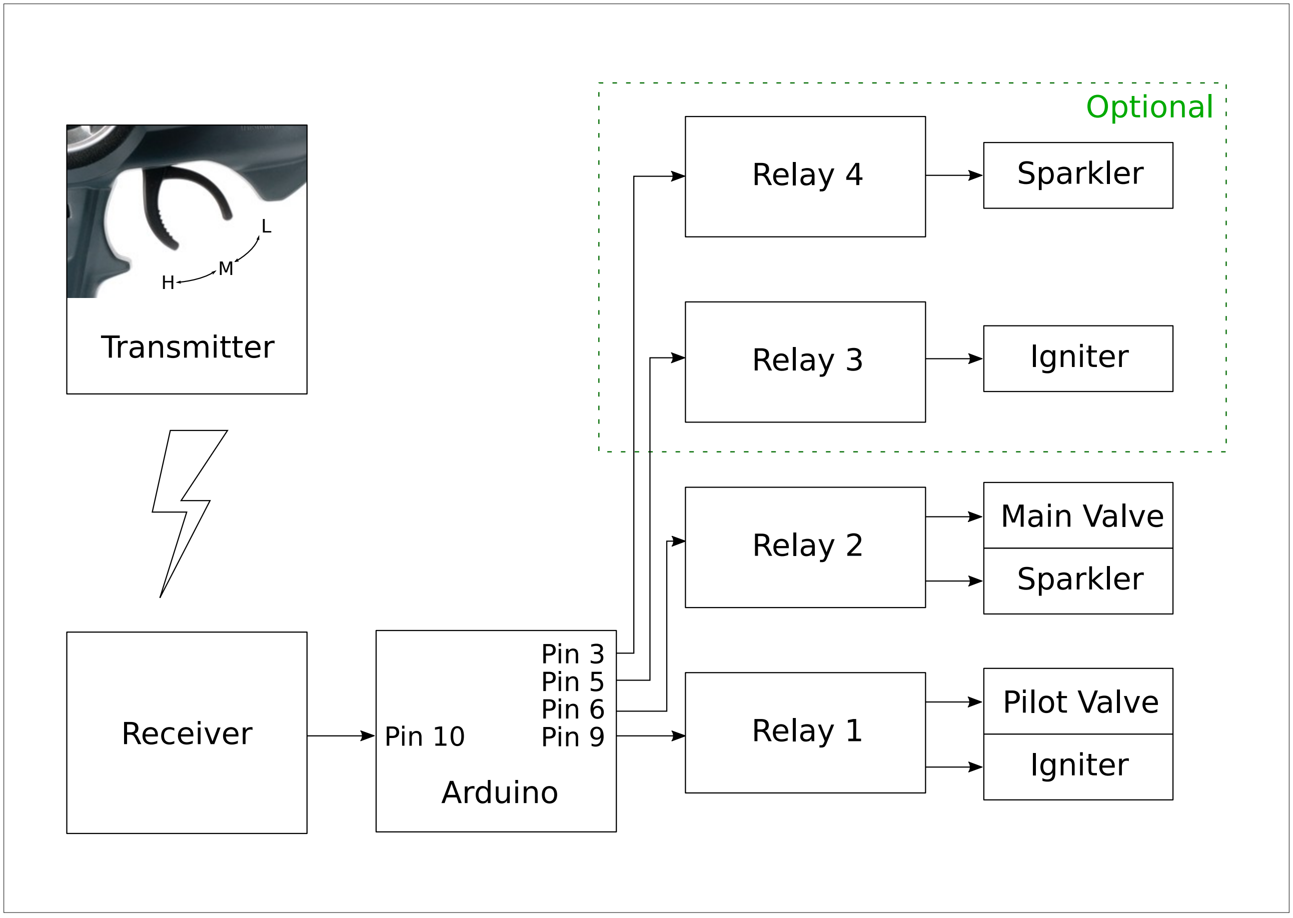

Figure 1: Hardware Overview

1.2 States

System Logical States				
State	**Relay 1 / Pilot Valve (Igniter)**	**Relay 2 / Main Valve (Sparkler)**	**Relay 3 / Igniter**	**Relay 4 / Sparkler**
Off	Off	Off	Off	Off
Idle	On	Off	On	Off
On	On	On	On	On

Table 1: Logical States

1.3 State Machine

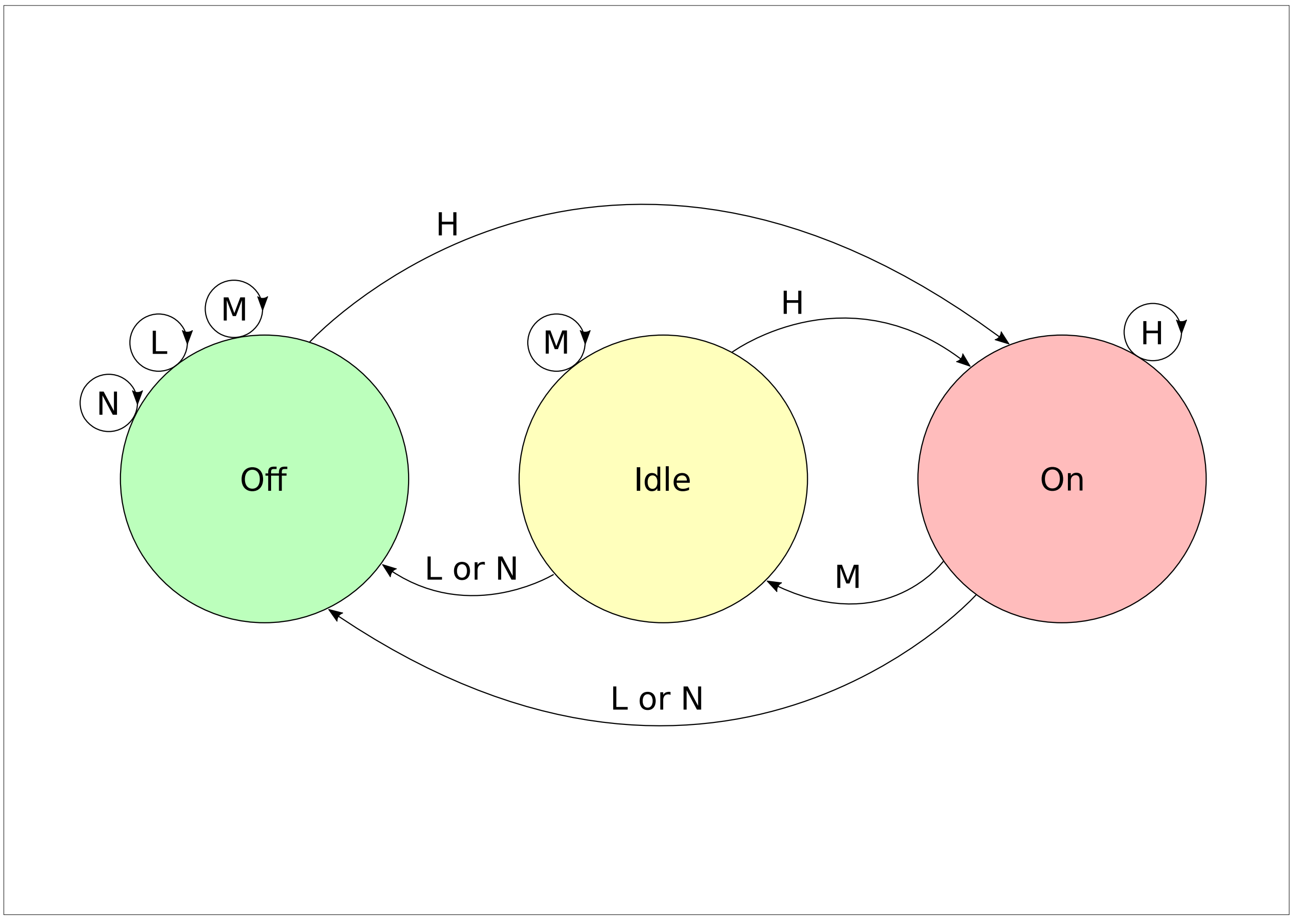

Figure 2: State Diagram

State Transition Table				
Controller Input		**Current → Next State**		
Name	**Code**	**Off...**	**Idle...**	**On...**
Low	L	Off	Off	Off
Medium	M	Off	Idle	Idle
High	H	On	On	On
No Signal	N	Off	Off	Off

Table 2: State Transitions

1.4 Input PWM Signal

Timings

The Arduino detects the input signal based on timing the period between the rising and falling edge of the input PWM signal:

Input Signal Timings		
Lower Bound (microseconds)	**Upper Bound (microseconds)**	**Signal**
0	500	NoSignal
500	1300	Low
1300	1700	Medium
1700	2500	High
2500	∞	NoSignal

Table 3: PWM Timings

Hysteresis

In order to prevent the input signal from rapidly changing when the input in near to a boundary, hysteresis is performed. The input signal length must be more than 100 microseconds outside the current input band in order for the input to change. This value can be changed by editing the `HYSTERESIS` definition in the sketch. Hysteresis is not applied when the current signal is NoSignal.

No Signal Handling

The Arduino watchdog timer is enabled to activate after 500 milliseconds. The rising edge interrupt resets the watchdog timer. Consequently if no signal at all is present, the Arduino will reset after 500ms. This value can be changed by editing the `TIMEOUT` definition in the sketch. After resetting, the Arduino initializes all outputs to Off.

If the input signal is present but the timing falls in the NoSignal ranges, the state machine switches to Off state immediately.

1.5 Output PWM Signal

Is handled by the Arduino Servo library.

2 Reference

2.1 Arduino Pro Micro Pin Out

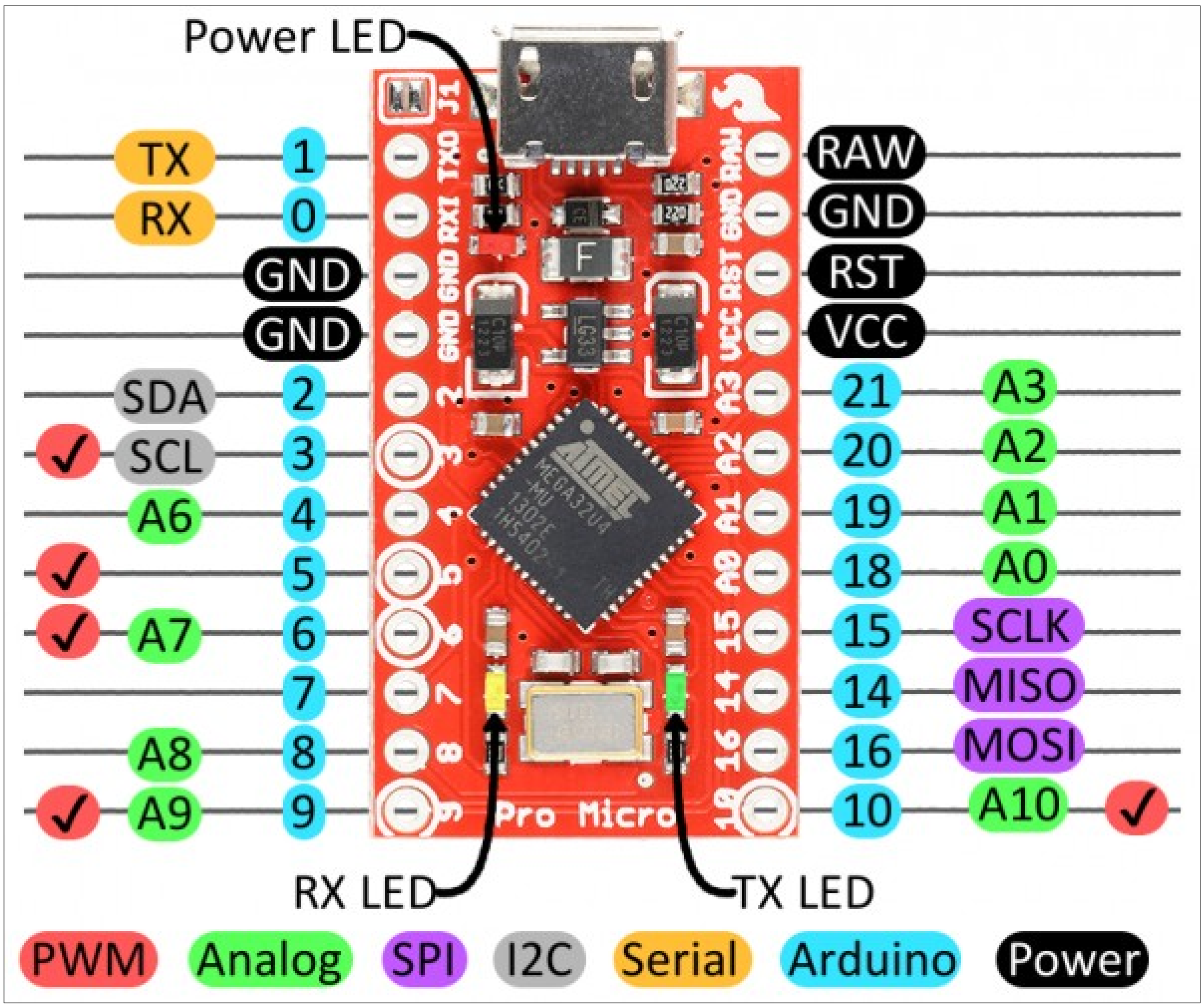

Figure 3: Arduino Pro Micro Pinout

2.2 Arduino IDE Setup

Permissions

Ensure you have added your user to the `dialout` group:

```
sudo gpasswd -a <user> dialout
```

Note: In order to activate the new group you must start a new login session.

Dependencies

To install required dependencies, click `Tools → Library Manager`. Search for the dependency and then click `Install`.

Note: If the Library Manager option is not present, it means your IDE version is too old.

The following dependencies are required:

1. `PinChangeInterrupt`
2. `Servo` *

Dependencies marked * should be installed by default with the Arduino IDE.

Target Board

Use `Tools → Board → Arduino Leonardo` as the target board type. This is a 5V board based on Atmega32U4, with the same logical pin mapping as Pro Micro.

Serial Monitor

The serial monitor is opened by clicking `Tools → Serial Monitor`. In older versions of the IDE, uploading a sketch will close the serial monitor.

Whenever the Arduino resets, the serial monitor will lose some of the first messages. This is because the USB device disconnects and reconnects, and the IDE must reopen it. This may cause the initial “Reset” message to be lost.

3. Testing

3.1 Direct Connection Test

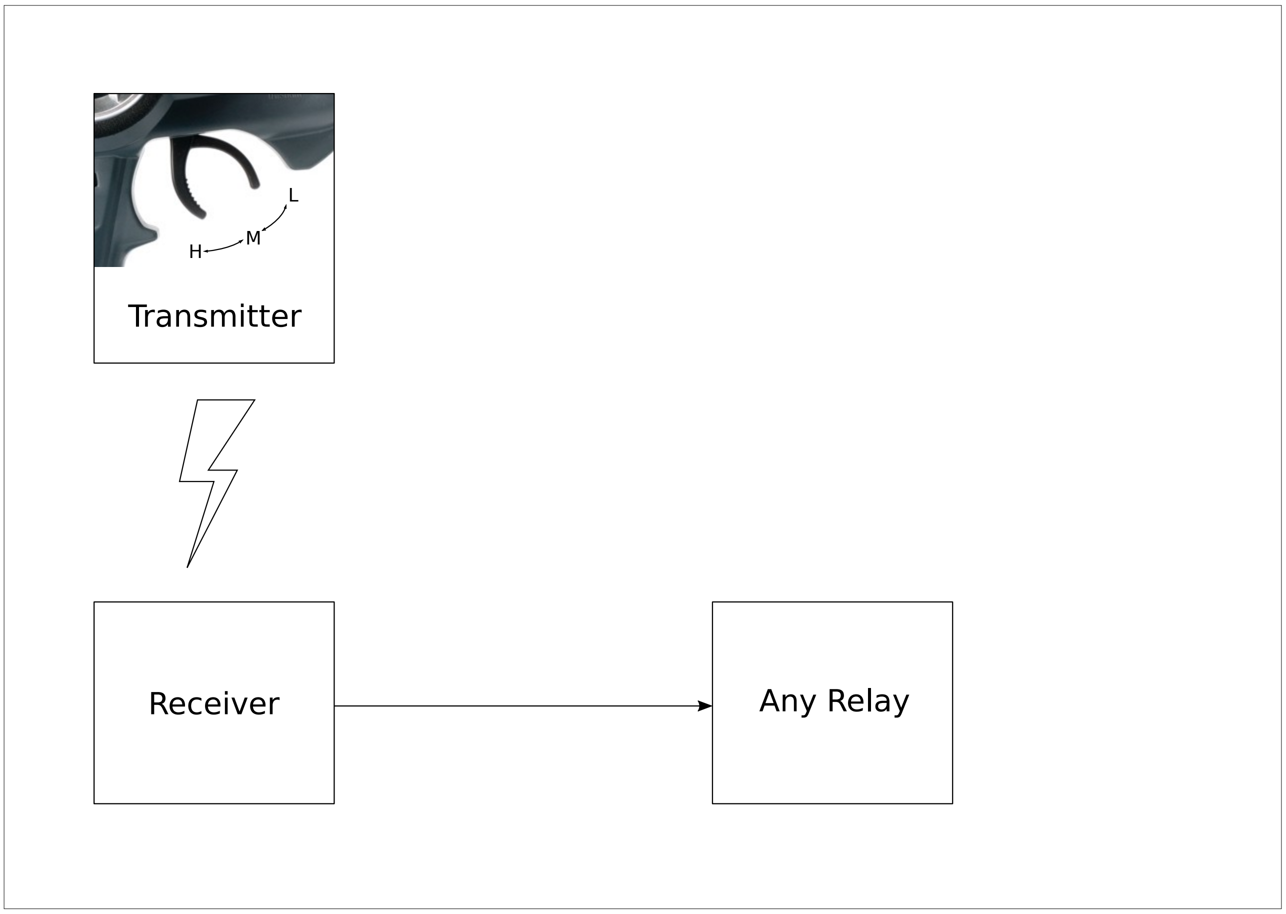

Figure 4: Direct Connection Setup

1. Connect Any relay directly to the output of the RC receiver.
2. Verify that moving the transmitter to ‘H’ turns on the relay.
3. Verify that moving the transmitter to ‘L’ turns off the relay.

3.2 Input Test

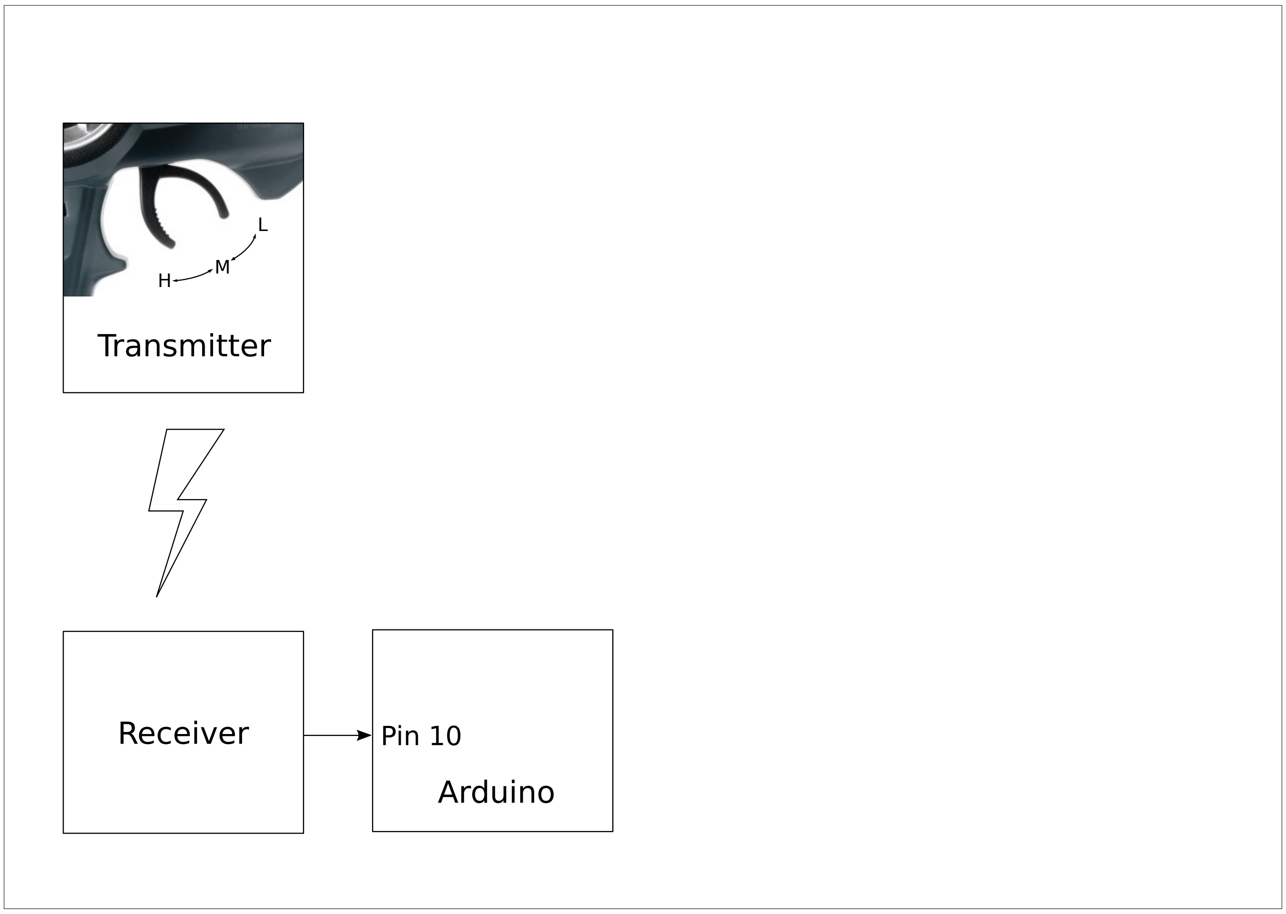

Figure 5: Input Test Setup

1. Program `passthrough` sketch to the Arduino.

2. Connect the RC receiver to pin 10 of the Arduino.

3 Open the Arduino Serial Monitor.

4. Verify that moving the transmitter trigger to ‘H’ causes the Arduino to output ‘Input: High’.

5. Verify that moving the transmitter trigger to ‘M’ causes the Arduino to output ‘Input: Medium’.

6. Verify that moving the transmitter trigger to ‘L’ causes the Arduino to output ‘Input: Low’.

7. Disconnect the receiver and verify that the Arduino outputs “Reset” and/or “Input: NoSignal” repeatedly.

3.3 Output Test

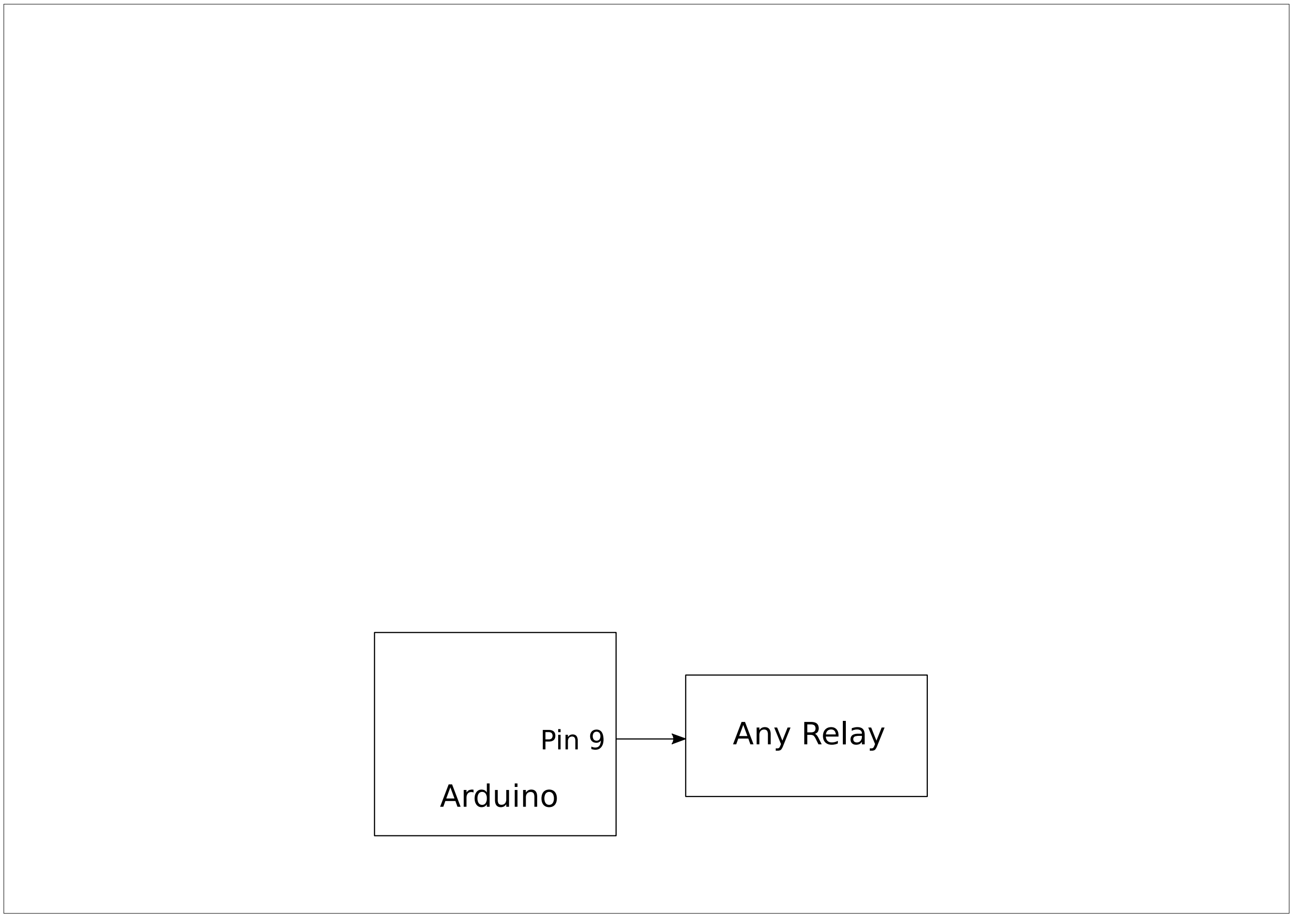

Figure 6: Output Test Setup

1. Program `File → Examples → Servo → Sweep` to the Arduino
2. Connect any relay to the Arduino pin 9.
3. Verify that the relay turns on and off approximately every 2.7 seconds. (2.7s on, 2.7s off.)

3.4 Pass-through Test

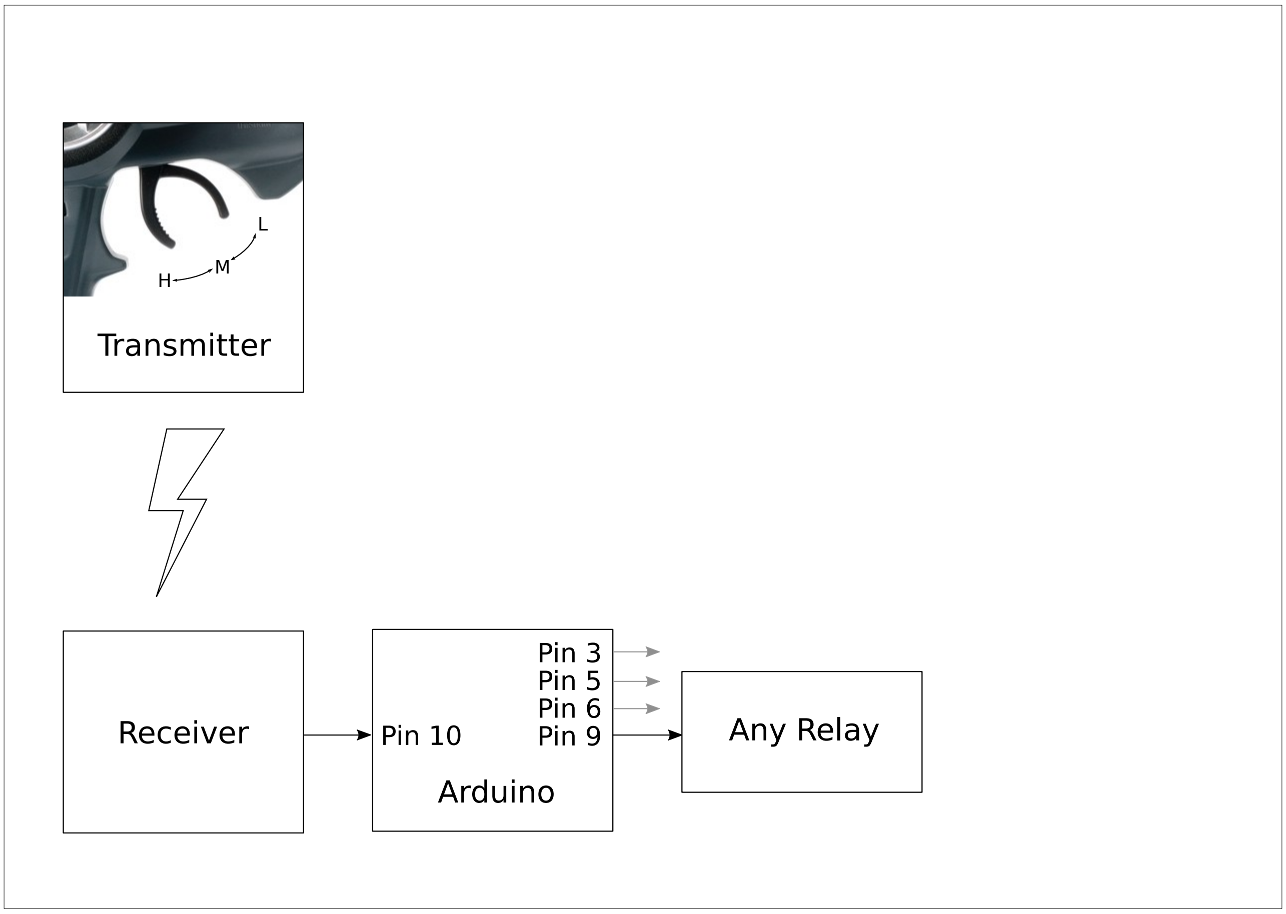

Figure 7: Pass-Through Test Setup

1. Program `passthrough` sketch to the Arduino.
2. Connect the RC receiver to pin 10 of the Arduino.
3. Connect any relay to pin 9 of the Arduino.
4. Verify that moving the transmitter trigger to ‘H’ causes the relay to turn on.
5. Verify that moving the transmitter trigger to ‘L’ causes the relay to turn off.
6. Move the relay to Arduino pin 3 and repeat steps 4 and 5.
7. Move the relay to Arduino pin 5 and repeat steps 4 and 5.
8. Move the relay to Arduino pin 6 and repeat steps 4 and 5.