Sensors connection guidelines

1) Infrared (IR) rangefinder:

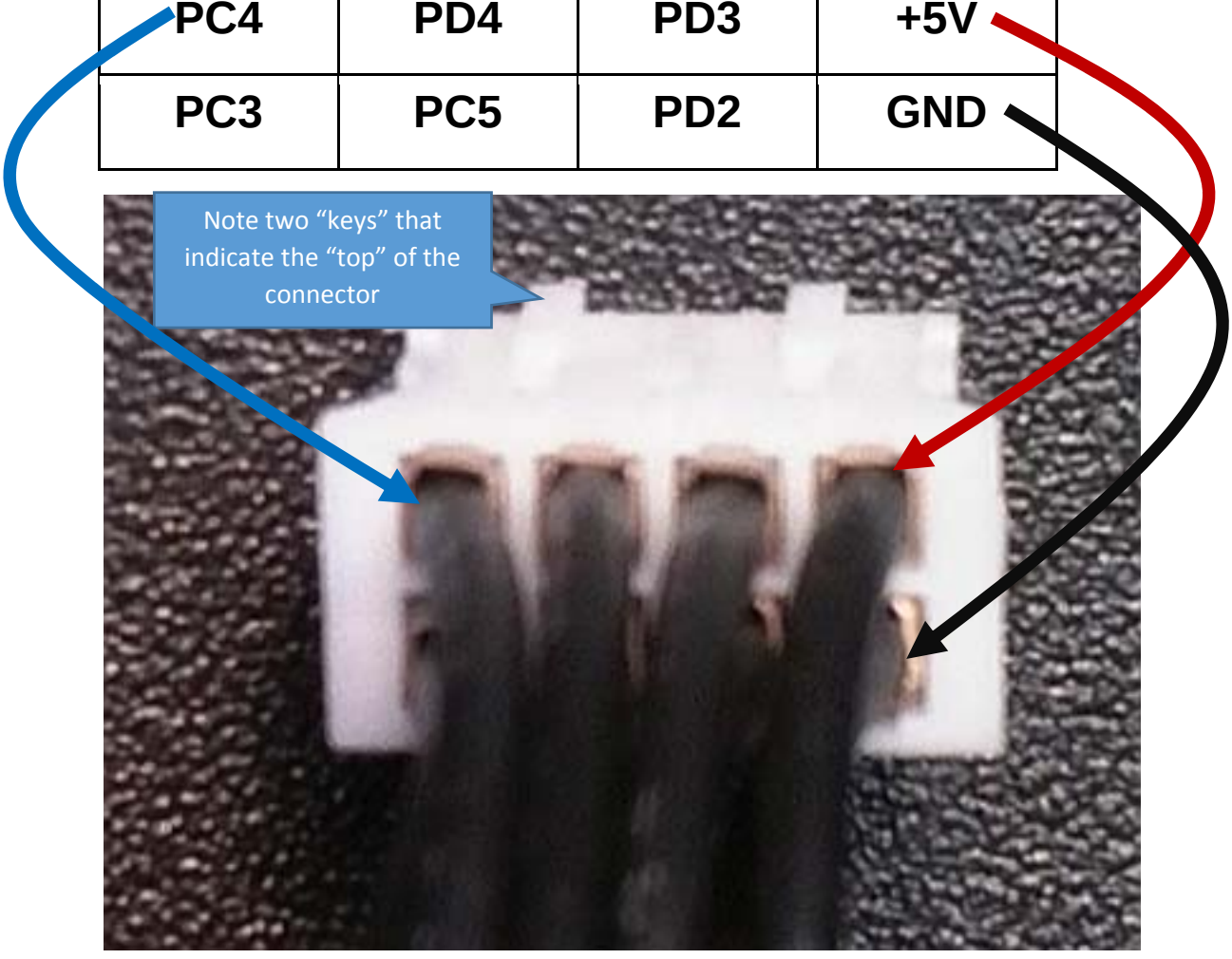
a. Analogue output (Max output voltage – 2.6V);
b. Must be connected to an ADC.

2) Ultrasonic (US) rangefinder:

a. Analogue output, i.e. needs an ADC;
b. Digital output (PWM), i.e. can use a GPIO.

Table 1: Expansion connector pinout (wire side view)

PC4	**PD4**	**PD3**	**+5V**
PC3	**PC5**	**PD2**	**GND**



Pins/wires description

#	Pin/wire name	Function/features	Use for the project	Colour of 1-pin connector
1.	PC4	GPIO, or ADC4, or **SDA**	IMU (SDA line).	Blue
2.	PC3	GPIO or **ADC3**	Connect an analogue sensor.	
3.	PD4	**GPIO**, or XCK, or T0	Connect a digital signal and work with it in either non-interrupt mode (polling) or pin-change interrupt mode.	
4.	PC5	GPIO, or ADC5, or **SCL**	IMU (SCL line).	Green
5.	PD3	**GPIO**, or OC2B, or **INT1**	Connect a digital signal and work with it in either non-interrupt mode (polling) or hardware (INT1) interrupt mode.	
6.	PD2	**GPIO** or **INT0**	Connect a digital signal and work with it in either non-interrupt mode (polling) or hardware (INT0) interrupt mode.	
7.	+5V	Poly-fuse protected +5V power	Power the sensors, 3.3V voltage regulator, etc.	Red
8.	GND	System ground	Must be connected to all peripherals.	Black

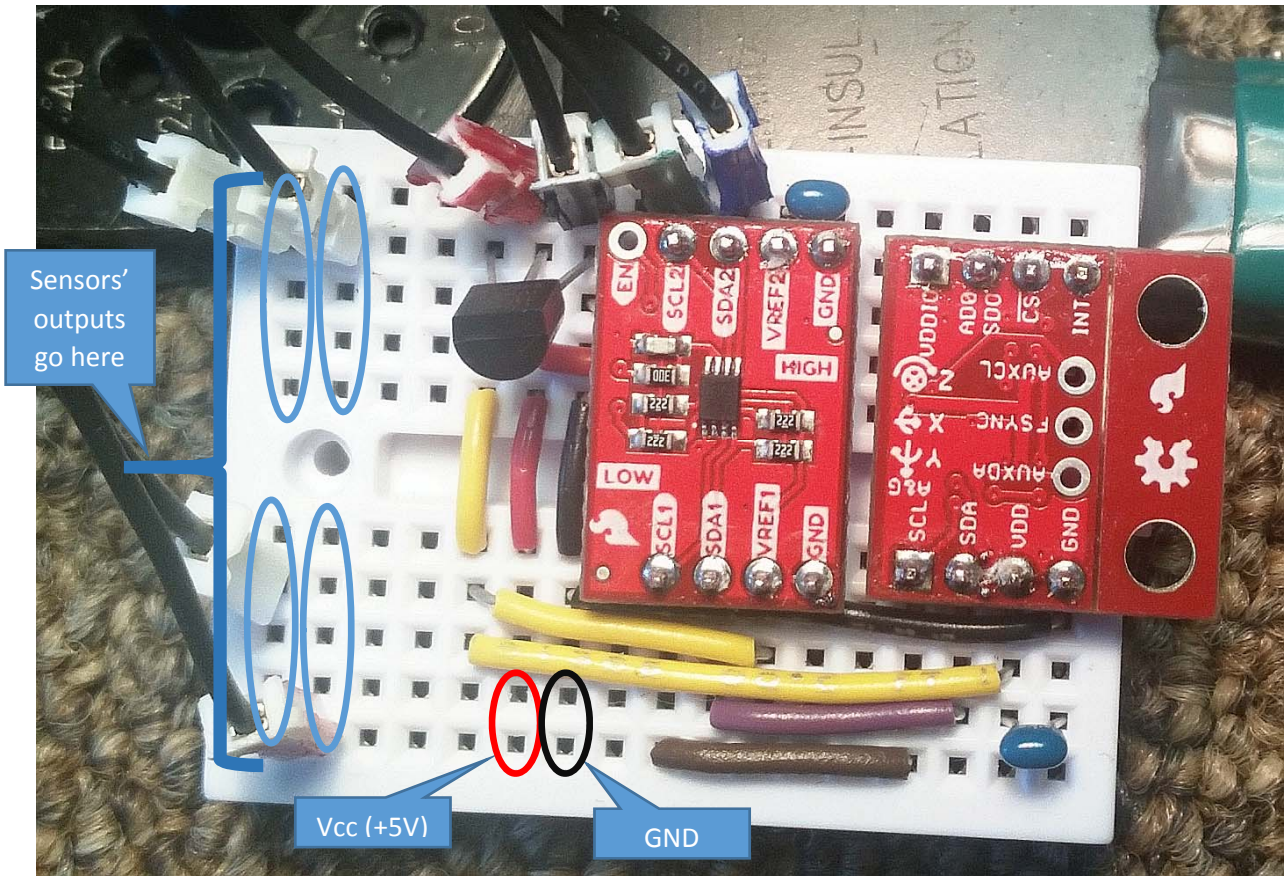


Figure 1: Sensors wiring

Programming instructions

1) Copy “init_290.c” and “init_290.h” to your working directory.
2) Add “#include "init_290.h"” at the beginning of your code.
3) In the “setup()” section:
4) Initialise the GPIO:
   gpio_init();

5) Initialise Timer1 to generate 20ms interrupts:
   timer1_50Hz_init(IRQ_enable);
   Now, TCNT1 will increment every 20ms/2500=8us. You can use it to measure the width of the US sensor response.

   If you wish to use Timer1 as a 20ms increment scheduler, you need to **enable the interrupt** and include in your code the ISR, something like:

```
ISR(TIMER1_CAPT_vect){ //system tick: 50Hz, 20ms
 time++; //aka 20ms ticks counter
 delay_ms+=20; // variable that tracks real time in ms. It could be reset independently from “time”
 flags.sample=1; // e.g. can be used in main() to start IMU sampling every 20mS
}
```

6) Initialise the ADC:
   adc_init (channel_number, enable_IRQ);
   As only ADC3 is available, use “adc_init (3,0);”

   If you wish to use the interrupt, you have to define the ISR. The code below averages ADS_sample_max samples and filters the values that are below the threshould:

```
ISR (ADC_vect){
        if (ADC_sample<ADC_sample_max){ // accumulating samples for averaging
                ADC_acc=ADC_acc+ADCH;
                ADC_sample++;
                return;
        }
                else {
                        ADC_val=(uint8_t)(ADC_acc/ADC_sample_max); // divide by # of samples
                        if (ADC_val<ADC_min) ADC_val=0;
                        ADC_acc=0;
                        ADC_sample=0;
                        ADCSRA&=~(1<<ADIE); // disable ADC IRQ until next sampling time. Re-enabling it in main() will update ADC_val.
                }
}
```