blog.thecell.eu

Weather station: Testing the entire windprobe setup

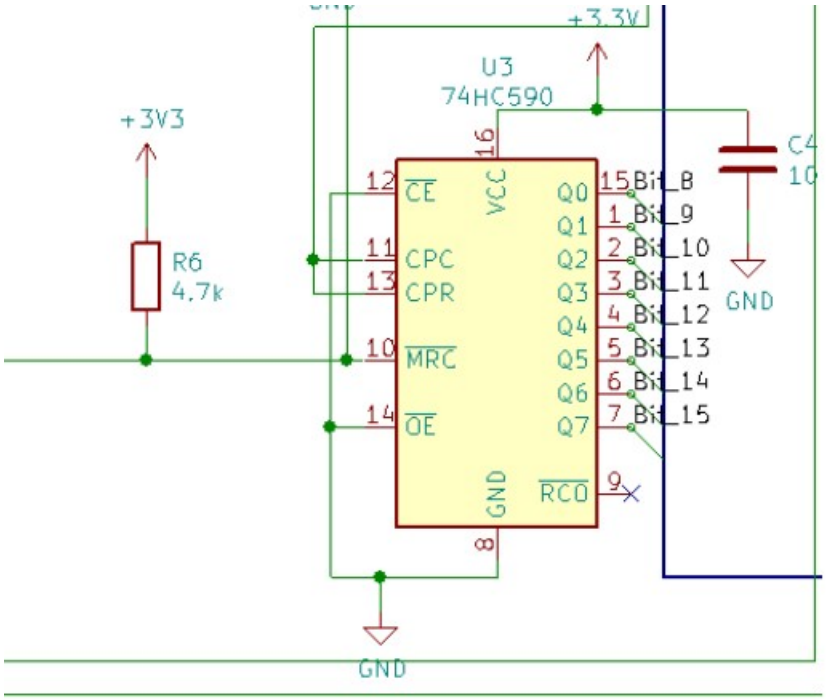
*Name **

4-5 minutes

On 25. February 2020 by TheCell

Hardware

This post summarises 2 sessions. We added a pull-up Resistor between the Raspberry and the probes counter reset line to remove uncertain behaviour in case the raspberry is booting up or is unconnected. You can get rid of the uncertain behaviour by just resetting the MCP and the counters but practices helps remembering :).



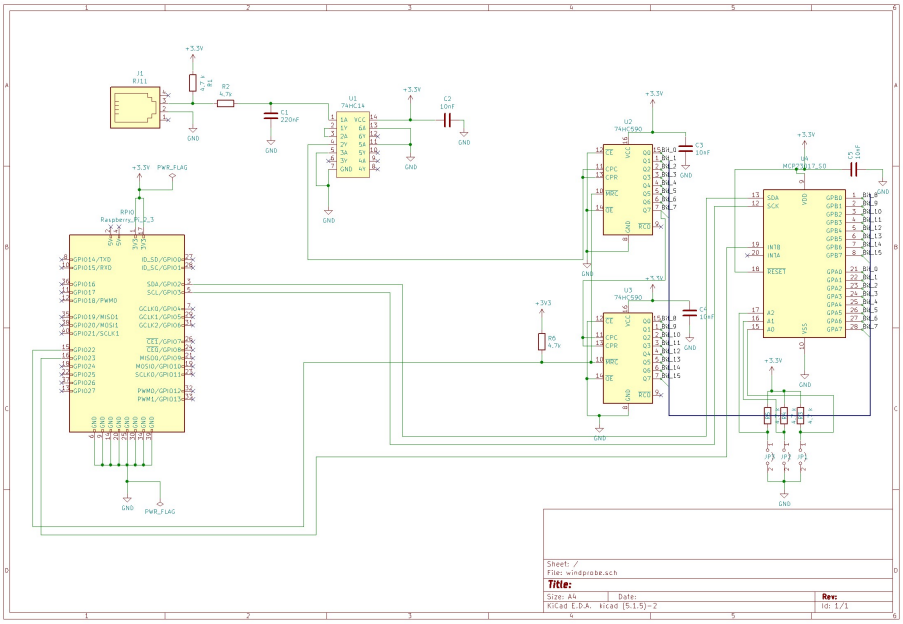
Pull-up resistor at the counter

Further we found out, that the windprobe we ordered sends a pulse every 1/2 turn and not as anticipated once per turn. This we will

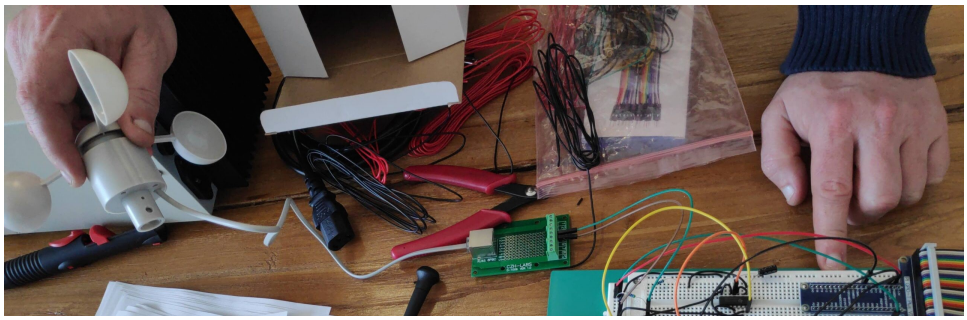
need to keep in mind for the counting and to make sure our filter is not cutting off the maximum windspeed we were aiming for.

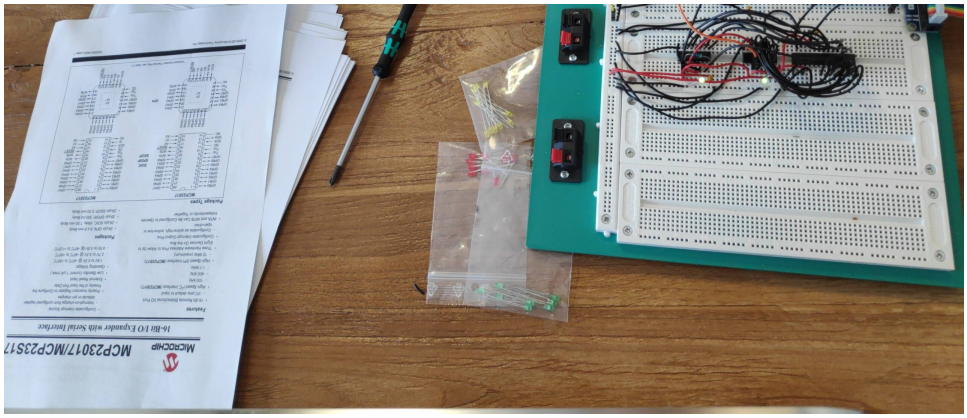
Brice introduced us to the basics of a bus. These limitations have to be kept in mind because there is signal bouncing happening in the bus. (Think of it kind of like water ripples if you build a physical water channel model of the bus and produce ripples on one end. The ripples bounce of the ends and bounce back to the producer, creating noise.)

- The bus needs pull-up Resistors at the end of the line.
- A bus can never be a star configuration (https://en.wikipedia.org/wiki/Star_network)
- All devices on the bus have to physically come one after another on the bus.
- The connection between the device and the bus should be as short as possible.



windprobe schema





final setup

Software

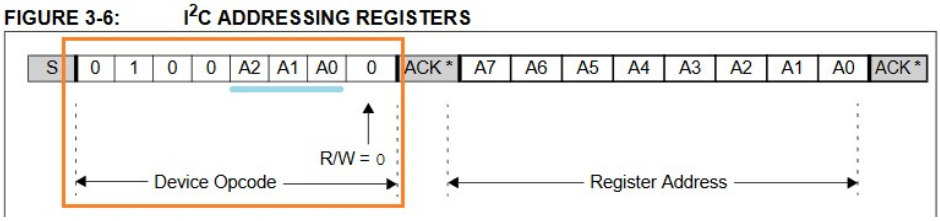
Our setup now is a wifi router connected to the internet, the raspberry connected via lan to the router and a laptop connected to the router aswell. We found the IP address of the raspberry on the router and connected to the raspberry via ssh (we use putty to connect (192.168.1.159:22 for us)).

To configure the raspberry to work via ssh check this raspberry manual page: https://www.raspberrypi.org/documentation/remote-access/ssh/ (Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)). Once we logged in on the raspberry we run 2 commands.

- sudo apt update
- sudo apt install i2c-tools

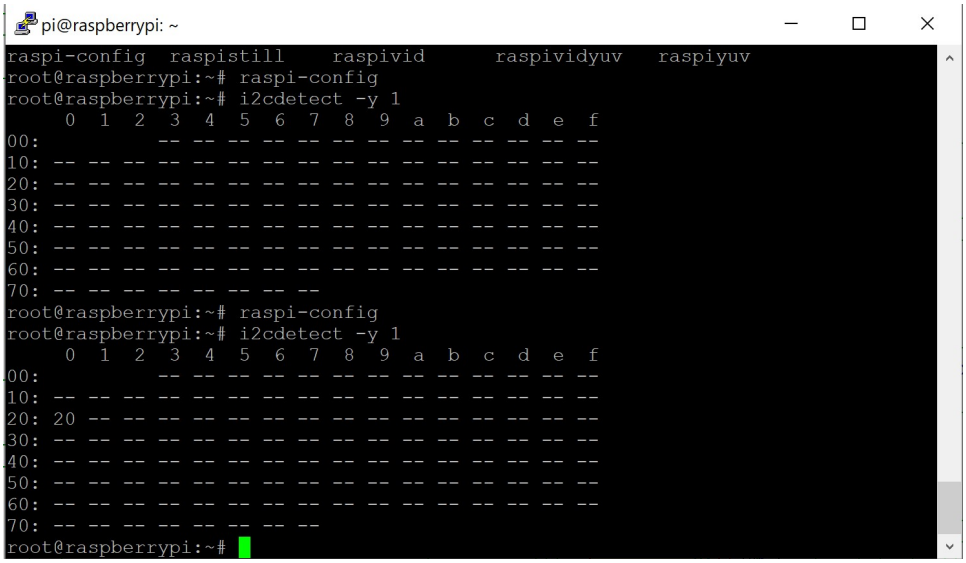
After that we have to enable i2cdetect to finally be able to read from our bus. To configure we write “**raspi-config**” navigate to 5 (Interface) and then I2C enable.

Now everything is set to check our bus. As you can see we have our device (here as 20) on the bus address 0x20 (column 0, row 20). The device is 20 because we configured the hardware pins to be 000 by grounding all 3 of them. The 4 predefined Bits are 0100. This means our Device is Binary 0**b**01000000 which in Hex is 0**x**20.



*The ACKs are provided by the MCP23017.

Device Opcode



Before enabling i2c and after

After a light reading in the manual we found how to read the bits. To program the MCP we will have to do a deep dive into the MCP datasheet (http://ww1.microchip.com/downloads/en/devicedoc/20001952c.pdf) to check if everything works we just use the raspberry with the command "**i2cget -y 1 0x20 0x12**" and "**i2cget -y 1 0x20 0x13**". See the i2c manual here (https://www.waveshare.com/wiki/Raspberry_Pi_Tutorial_Series:_I2C and here https://linux.die.net/man/8/i2cget)

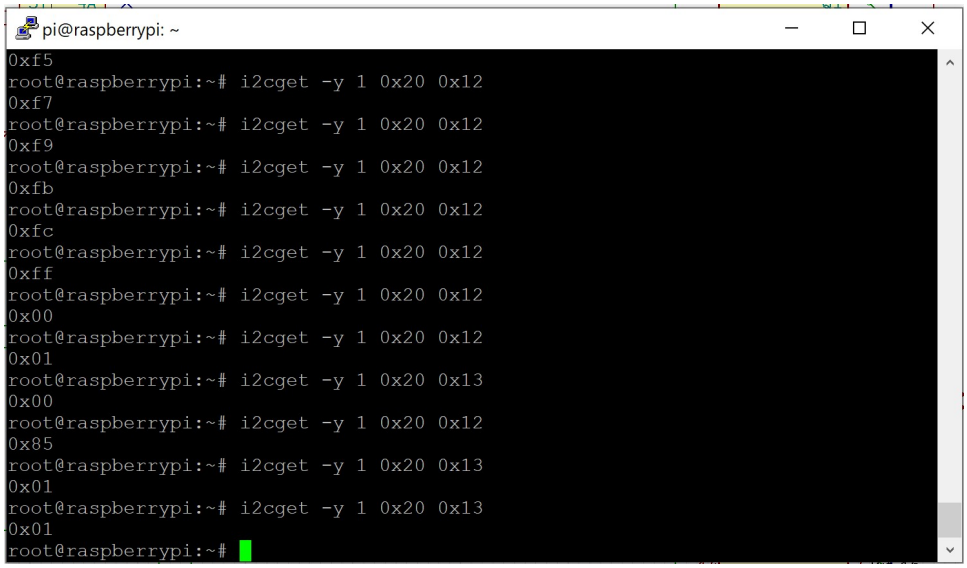
- i2cget: read bytes
- -y: Disable interactive mode
- 1: Indicates the number or name of the I2C bus to be scanned.
- 0x20: Address of the MCP I/O extension
- 0x12: Addres in the MCP to read the first 8 bits, see next screenshot
- 0x13: Addres in the MCP to read second 8 bits, see next screenshot

TABLE 3-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE GPIO PORTS (BANK = 0)

Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IPOLA	02	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
IPOLB	03	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000

GPINTENA	04	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPINTENB	05	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPPUA	0C	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPPUB	0D	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPIOA	12	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
GPIOB	13	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	14	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
OLATB	15	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

Datasheet GPIO Address



Reading form the i2c multiple times

As you can see by turning the probe and reading the registers the number is increasing. Once we pass 0xFF it is reset to 0 and we can read on the second Bit bank.

Github Link: https://github.com/TheCell/Weatherstation