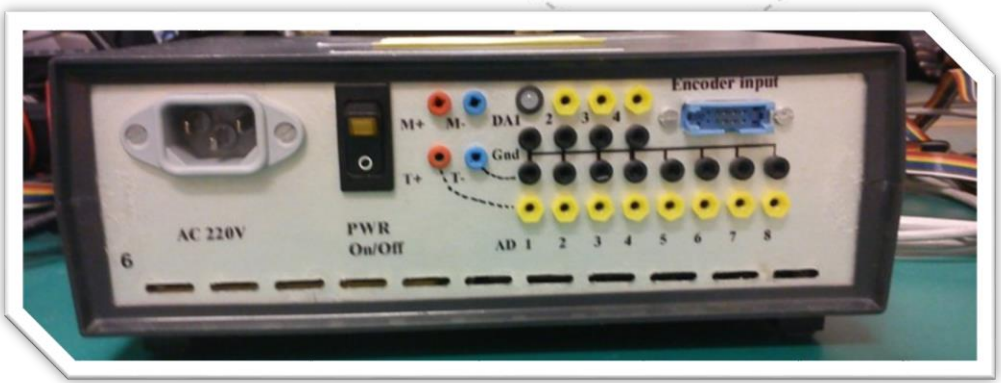
TTK4155

Industrial and Embedded Computer Systems Design

NTNU – Trondheim
Norwegian University of
Science and Technology

Lab lecture 8

- I^2C bus
- Motor controller box
- Solenoid



Exercise 8: Controlling motor and solenoid

- In this exercise, you will
  - Connect the motor controller box to Arduino and I/O board
  - Connect the ping pong board to the motor controller box
  - Use the Atmel I^2C driver to control the DAC on the I/O board
  - Create a motor control driver and test with joystick input
  - Create a position controller using feedback from the quadrature encoder
  - Connect the solenoid and use a button on the joystick to fire

Motor box interface

Motor box interface

- Simple control pin interface.
- See ‘Motorbox User Guide’ on Blackboard.

Connector	Pin	Name	Comment
MJ1	2	Gnd	Common ground
MJ1	3	!OE	Active low output enable of encoder
MJ1	4	!RST	Active low reset of encoder
MJ1	5	SEL	Select Hi/Lo byte of encoder
MJ1	6	EN	Enables motor
MJ1	7	DIR	Sets the direction

Connector	Pin	Name	Comment
MJ2	2	Gnd	Common ground
MJ2	3 – 10	DO0-7	Output 0-7

Connector	Pin	Name	Comment
MJEX	1	DA1	Analog input
MJEX	2	Gnd	Common ground

MJ1

GND	!RST	EN		
2	4	6	8	10
1	3	5	7	9
	!OE	SEL	DIR	

MJ2

GND	DO1	DO3	DO5	DO7
2	4	6	8	10
1	3	5	7	9
	DO0	DO2	DO4	DO6

MJEX

GND							
2	4	6	8	10	12	14	16
1	3	5	7	9	11	13	15
DA1							

Basic control of motor

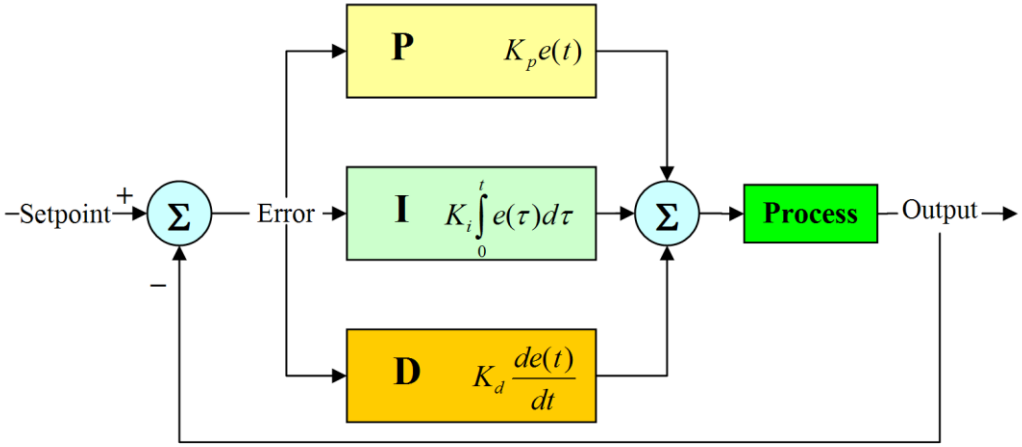
- Set motor “speed” by adjusting DAC voltage
- Select direction with DIR pin
- Set EN pin high to enable motor

Read motor encoder

- Internal 16 bits counter
- !OE pin activates output of encoder counter
- SEL selects either high byte (0) or low byte (1)
- !RST resets the counter
- See guide for more detailed steps.

Position regulator

- For example a PID regulator



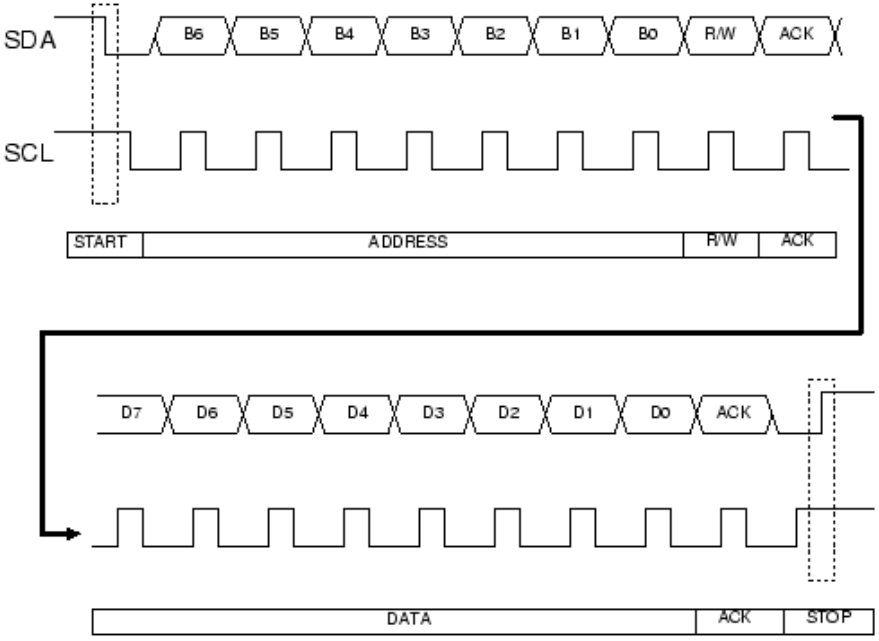
- The quadrature encoder will give an indication of the motor position.
- The error is the difference between the indicated position and the desired position (from the joystick).
- Calculate the necessary values P, I and (D).
- Sum all values and apply to motor.

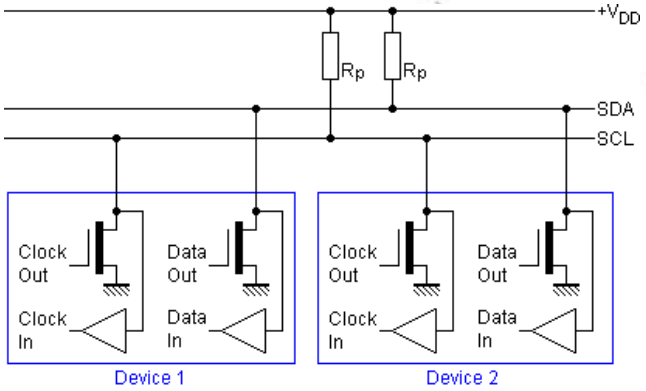
Inter-Integrated Circuit

Feature	SPI	CAN	I2C
Same PCB	Yes	Not ideal	Yes
Between PCBs	Not ideal	Yes!!!	Not ideal
Multi master	Requires hacking	Yes	Yes
Multi slave	Support through multiple CS or daisy chain	Yes	Yes
# wires	4 or more	2	2
Speed (throughput)	Variable (possibly high)	Mid	Slow

DAC interface

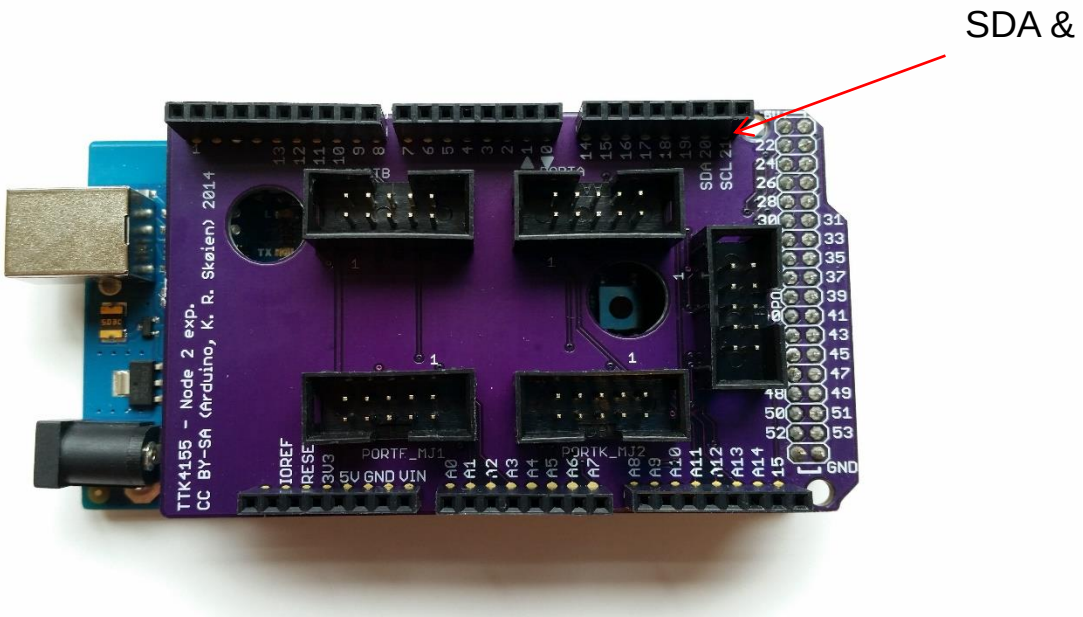
- Use I²C to interface with the DAC
- I²C is a two-wire protocol, clock line and bi-directional data line
- Master/slave configuration
- Fairly simple protocol:





Atmega I2C driver

- Can be found on Blackboard under TWI driver.
- See also application note AVR315 how to use the driver.
- You can modify it as you wish, write your own driver or use some other driver. As long as you



Using the MAX520 DAC

- Fairly straight forward, Consult the data sheet

First 4 address bits factory set: 0101

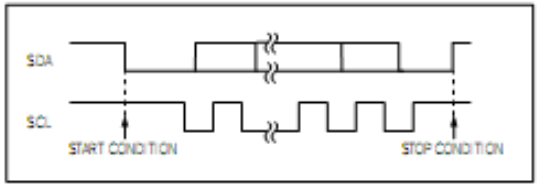


Figure 4. All communications begin with a START condition and end with a STOP condition, both generated by a bus master.

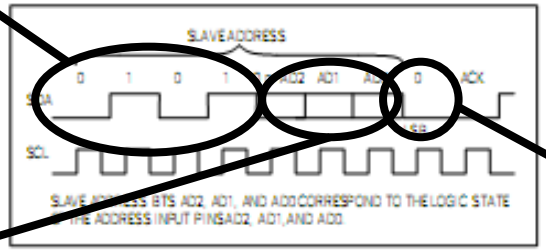


Figure 5. Address Byte

3 bits set by connecting AD0-AD3 to GND/VCC

Last address bit is zero

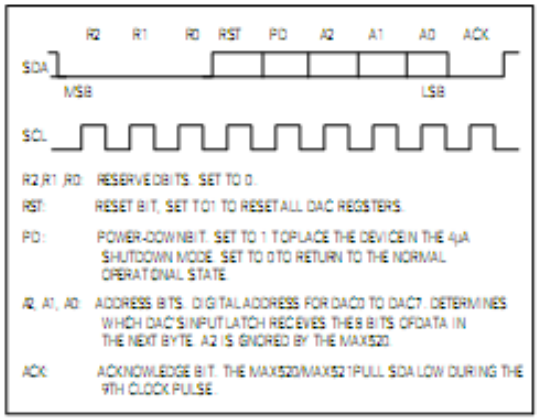


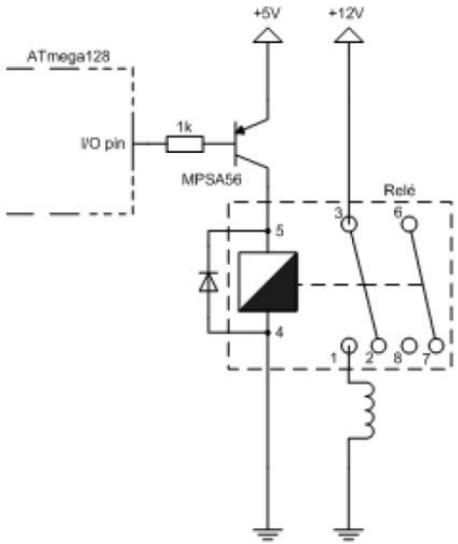
Figure 6. Command Byte

The address is followed by A command and a output byte

NTNU – Trondheim
Norwegian University of Science and Technology

Solenoid

- The solenoid requires 12V (at least) to throw a punch.
- A relay have to be used.
- The AVR I/O pins can not supply the necessary current to drive the relay.
- A transistor is used.
- Protective diode at the relay input.
  - Free-wheeling diode.



Extras….

- Make sure game is working.
- Keep extras inside embedded realm.
- Some ideas;
  - Well documented code e.g. UML.
  - Dual buffer and creative use of OLED e.g. animations etc.
  - Online tuning of PID controller.
  - Use of cell phone's sensor e.g. gyro, accelerometer (requires extra comm. module)
  - Explore & use modules from USB card e.g. buzzer, CAN etc.
  - More ideas => at the end of lab compendium text.
- In short, extras can be simple and still earn points.

Questions?