Hochschule Konstanz
Department of Computer Science

Prof. Dr. Ralf E.D. Seepold
Ubiquitous Computing Lab

Degree: Applied Informatics (AIN)
Lecture: Ubiquitous Computing

Laboratory 1: Arduino

Objective: write the text here

1 Basic working with Edison IO, PWM and Serial

1.1 First Task to be done

Intel Edison is a Small Embedded System with basic functionality like IO Pins; some of them are with PWM functionality, data bus (SPI, I2C, UART).

The Standard OS in Intel Edison is Linux Yacto although other OS are also available

Intel Edison supports a variety of programing and scripting languages like C/C++, note.js, python ... etc.

Here some web links with tutorials and examples how to use System Buses on Intel Edison. The Examples will use C++.

https://iotdk.intel.com/docs/master/upm/

https://iotdk.intel.com/docs/master/mraa/

https://iotdk.intel.com/docs/mraa/v1.8.0/edison.html

https://github.com/intel-iot-devkit/mraa/tree/master/examples

How to use the Motor Driver.

In Edison Cars is used a small Motor driver (TB6612FNG). To interact with this motor driver we have to understand the Control Function of the driver.

Interpretation:

If two inputs are High, the motor will break. If the first input is Low and the second is High, the first output will always be Low and the second output will have the value of the PWM.

If the first input is high and the second is Low, the first output will have the value of the PWM and the second output will be low. If both inputs are LOW and PWM will have a high the driver will stop.

The driver will be set on standby, if you set the STBY to Low; this will save Battery.

We also have to declare and configure some pins. Following Pins will be used for the motor driver. Motor A (pin 0 (pwm), pin 45, pin 46) Motor B (pin 14 (pwm), pin 47, pin 48) and STBY pin 15.

As shown in the links above, to initialise a pin we have to:

Declare it and initialise mraa::Pwm* pwm_pina = new mraa::Pwm(0);

Check for correct initialisation if (pwm_pina == NULL) {std::cerr << "Cant create mraa::Pwm object, exiting" << std::endl; return MRAA_ERROR_UNSPECIFIED;}

by PWM pins enable the PWM Mode and check if (pwm_pina->enable(true) != MRAA_SUCCESS) {std::cerr << "Cannot enable PWM on mraa::PWM object, exiting" << std::endl; return MRAA_ERROR_UNSPECIFIED; }

by IO pins you have to set the direction if (stby_pin->dir(mraa::DIR_OUT) != MRAA_SUCCESS) {std::cerr << "Cant set digital pin as output, exiting" << std::endl;return MRAA_ERROR_UNSPECIFIED; }

and to set some value on the IO pins use the function, write stby_pin->write(0);

PWM also known as pulse width modulation is used to control power supplies, for example, Motors and LEDs. The term duty cycle is used for defining the proportion of "high" or "on" time to the regular interval "period" and is usually expressed in percent from 0% to 100% (0.0 to 1.0) where 100% (1.0) is fully on.

For our implementation in Edison this means that we need a duty cycle with the values between 0 and 1 and a period in nanoseconds or milliseconds.

duty cycle	float duty_cycle = 0.0
period	int period = 1
to set certain pwm on a pin	pwm_pina->config_percent(period, duty_cycle)
to write	pwm_pina->write(1.0);
and free res when you don't need them	delete pwm_pina

1.2 Useful information. (not relevant for the lab)

After we understood the last steps, it is necessary to create a connection to our wireless connection. In this example we will use an Xbee module that communicates via UART (Serial) with the Edison.

The principles are similar to all modules that communicate via UART.

Declare it and initialise the device	mraa:Uart* dev = new mmraa::Uart(0)
set the baud rate	dev->setBaudRate(115200)
set the parity	dev->setMode(8,mraa:UART_PARITY_NONE,1)
set the flow control	dev->setFlowcontrol(false,false)
and finally send some text	dev->writeString("Hello NOW")
and free RES when you don't need them	delete dev

same is also possible for writing Char arrays dev->Write(buffer, sizeof(buffer))

where buffer is char buffer[] = "Hello Bufer!"

For all supported functions check the references of the mraa lib.

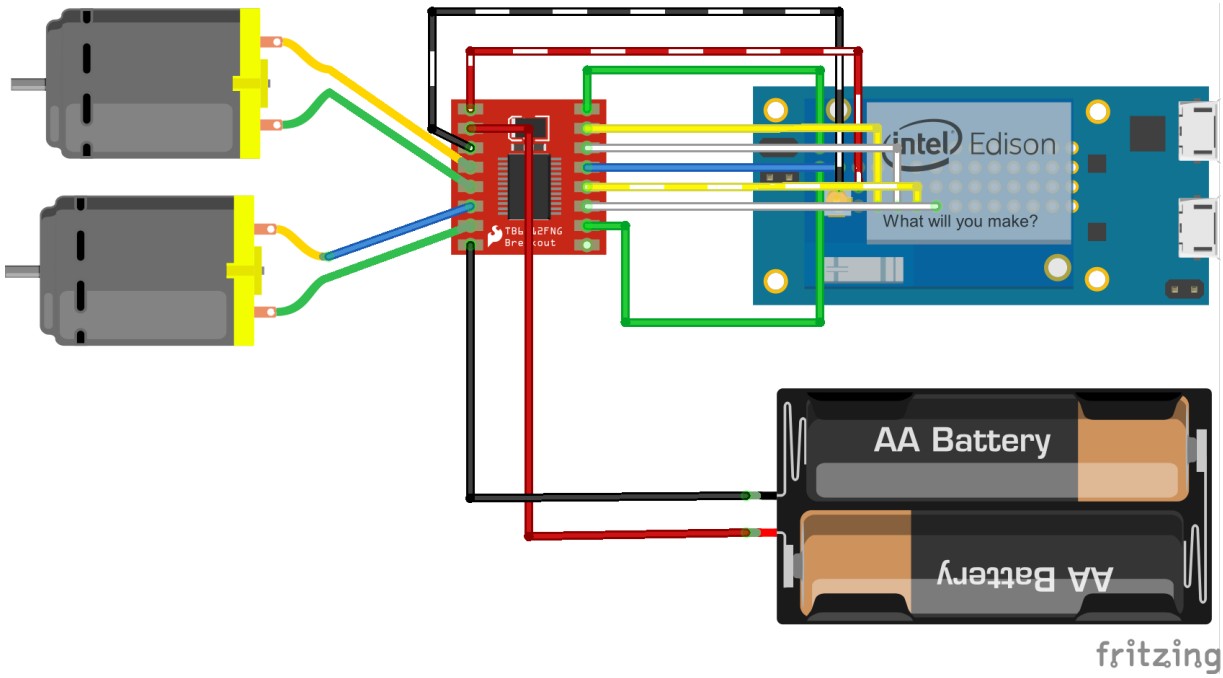
https://iotdk.intel.com/docs/master/mraa/

Now having a communication with the module we can configure it and use it with the AT commands and sending data.

Now you got all the necessary tools for work. Enjoy coding!

1.3 System wiring and Motor driver

The following draft shows the wiring inside the car.



The next table shows the functionality of the Motor driver

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

1.4 Task: Controlling

Using the descriptions and instructions of above. Design a remote controlled system that drives the car. The main functionality is driving back and forwards, left and right. As remote operator you can use a PS4 Controller, an Web Interface or a App on a smartphone.

Describe why did you use the chosen remote controller. What are the advantages and disadvantages.

Describe how does the communication whit the system works. How did you read the file, sockets or data stream.

Please upload the Design/Concept, and the description of the communication on model. Do not add unnecessary code to the report.