PSR - Semestral work: Motor control.

Leonid Tulin, Ivanov Mikhail

January 2019

1 Introduction

The goal of the semestral work is to create a digital motor controller. Program will control the position of the motor according to the set-point given by the position of another motor, moved by hand (steer-by-wire). The set-point will be transferred between the two motor controllers using UDP messages. The actual state of the controller and its history will be published as live graphs over the HTTP protocol.

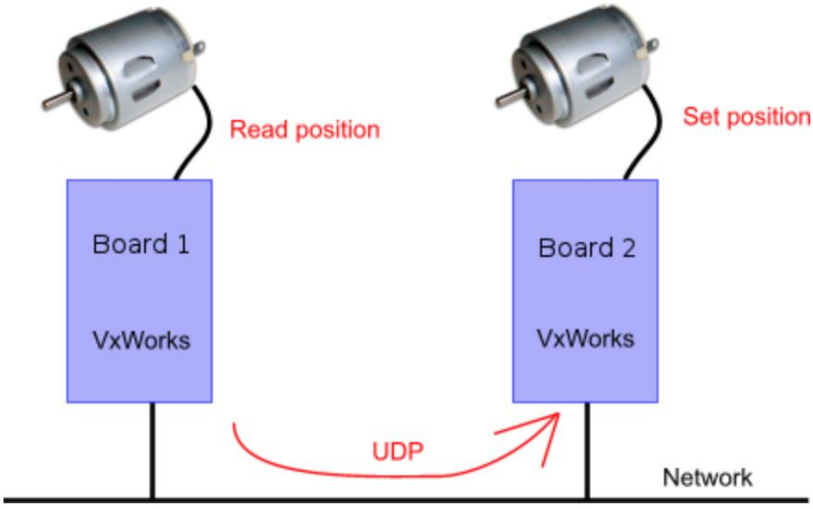

Figure 1: Schematic description

2 Application Programming Interface (API)

Both **server** (board that reads and sends its motor position (target)) and **client** (board that changes its motor position based on target position) has individual API, however they have some mutual interface.

2.1 Common API

2.1.1 Functions

void motor(void)	Main motor function to initialize motor, communication and receive data.
void irc_init(void)	Initialization of motor control and interrupts.
void irc_disable(void)	Disabling of interrupts.
void irc_isr(void)	Interrupt Service Routine (ISR) – reads new position and release semaphore irc_sem.
void irc_print_status(void)	Routine to update current position based on gray code, waits for irc_sem.

2.1.2 Variables

SEM_ID irc_sem	Semaphore ID that's released by ISR irc_isr and taken by irc_print_status.
volatile int irc_a, irc_b	Integers to store grey code received from motor in interrupt handler.
volatile char last[2]	Char string to keep last motor position grey code in char[] .
volatile long int k	Board's motor position (signed integer). Initialized as 0.

2.2 Client API

2.2.1 Functions

void contr(int dir, int speed)	Changes FPGA bits to control motor rotation. Takes direction (1 – clockwise, 0 – counter clockwise) and rotation speed (PWM duty cycle).
void regulator()	Calculates direction and rotation speed (PWM duty cycle) based on target and motor position and calls void contr(int dir, int speed) with respective arguments. PI controller is implemented. Maximum PWM value is defines as 5000.
void send_Respond(void)	Sends set board (client) position to server.

2.2.2 Variables

volatile int konec	Target board's motor position. Initialized as 0.
volatile char now[2]	Char string to keep current motor position grey code in char[].
volatile int lastError	Keeps last error (used at I part of PI controller). Initialized as 0.

2.3 Server API

2.3.1 Functions

void send_Lenya(void)	Sends target position to client.
void recieveFrom(void)	Receives client position (used for graph plotting).
void updateData(void)	Updates data arrays for graph plotting.
void www(int* s, int *dataN, int argc, char *argv[])	Function for web-graph plotting. Takes pointer to socket s (for proper socket closing out of www function), current data time pointer dataN and arguments for socket.
void newRequest(int newFd, int *dataN)	Supportive function of void www(…). Process new requests and generates web page as well as SVG graph based on data arrays and time dataN.

2.3.2 Variables

volatile int dataN	New data writing position (used for data arrays update). Initialized as 0.
int s, sockd2	Sockets for UDP communication. Set global for proper socket closing.
int pM[1024]	Data array to store server position at time dataN
int pS[1024]	Data array to store client position at time dataN
int pE[1024]	Data array to store error at time dataN
#define lenyaAddr "X.X.X.X"	Address of client board

3 Web server

Server board runs web server available at its address. The page includes short legend – x axis represents time (in seconds), y – position (in steps). Blue curve is used for server position plotting, green for client and red represents error (error = server – client).

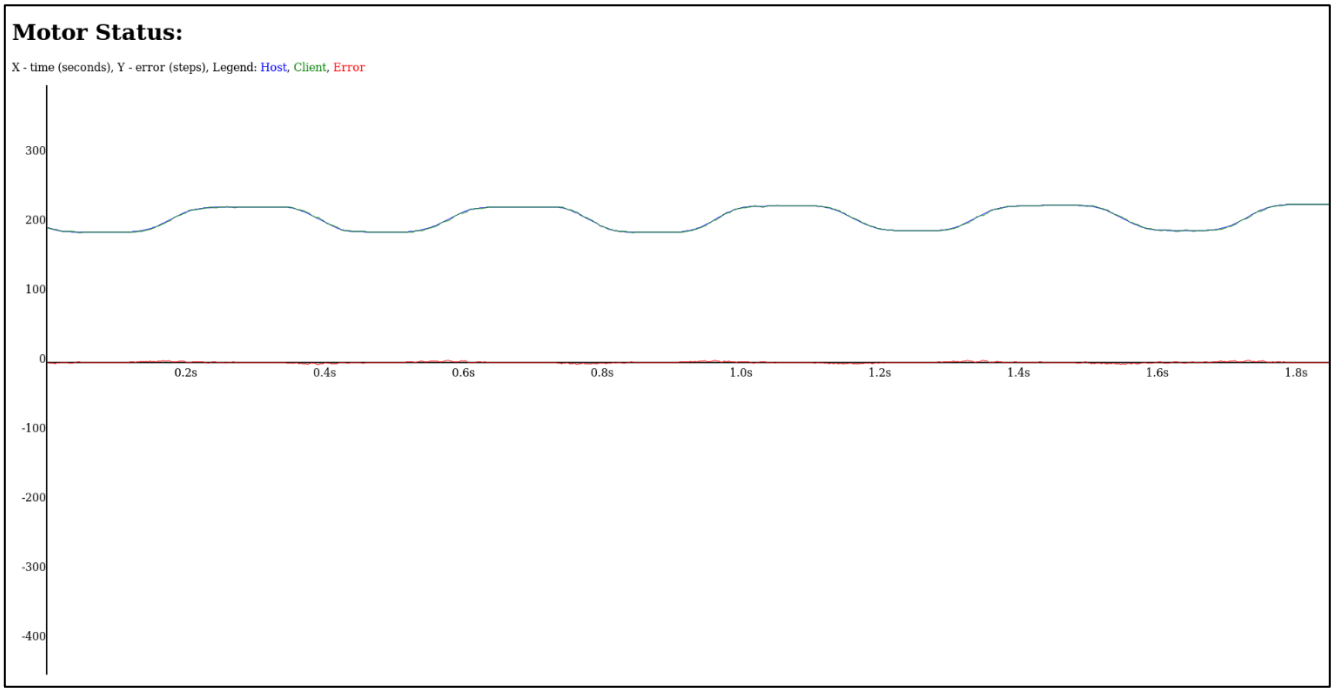

Figure 2: Web-server for plotting

4 Data Flow

Data flow is shown on Fig.3. Boxes represent board functions, black lines – in-board communication, red lines – out of board communication. Each function stated as func(freq, priority) and short description, where freq indicates its frequency either in board timer cycles or in written form (e.g. every request). Priority stands for VxWorks priority level, where lower value implies higher priority.

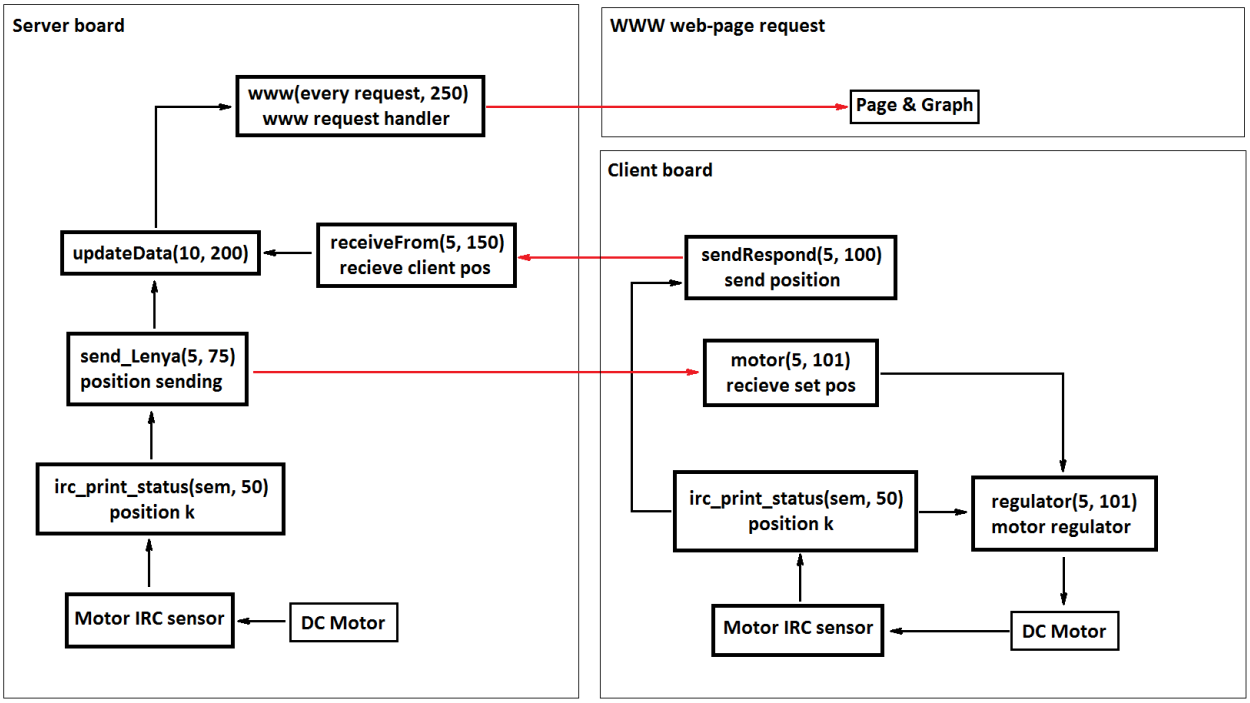

Figure 3: Data Flow

5 Compiling and Running

For proper project compiling it's necessary to compile project as Kernel Downloadable Module for ARMARCH7_SMP and include additional paths for builder/compiler (see below). It's also needed to indicate client and server addresses at respective variables/functions.

```
-I$(WIND_BASE)/target/config/xlnx_zynq7k
-I$(WIND_BASE)/target/config/lite5200b
```