

Embedded and Real-time Systems

Term Project Assignment

The goal of the project is to develop an application for controlling a punch press and deliver a report that provides argumentation about real-time properties of the application.

Application

The application shall punch holes centered at the following positions provided in the file **punches.in**. Then the head should return to position [0; 0] and stop there. To move the head you are required to implement a [PID controller](#).

The application should be written for STM32F4 board using one of the methods done in previous labs. The [punch press](#) simulator runs on another STM32F4 board providing output through serial port and feedback via dedicated pins. The serial port output can be viewed in provided visualizer application.

The device that the application drives is described in the last section of the document.

Report

The report that should be submitted along with the application shall overview the structure of your system and should provide an argument why the timing requirements of the system are met. In particular your report should cover the following:

- What are the timing requirements of the system and why.
- What is the structure of your implementation (IRQs, tasks, communication among them, synchronization, ...)
- How do you model the system from the schedulability perspective (tasks and their parameters) and why this reflects your implementation?
- How did you estimate the worst-case execution times and why do you think they represent the worst-case.
- How did you perform the schedulability analysis? Explain what the results of the schedulability analysis mean.

Submission

The term project shall be submitted by the **end of August**. The submission shall contain the following:

- Source code for the application
- Makefile with targets for compilation and flashing
- Report

Your term project will be evaluated and given 0-60 points. The final grade will be based on the points for the term project and points for the examination test.

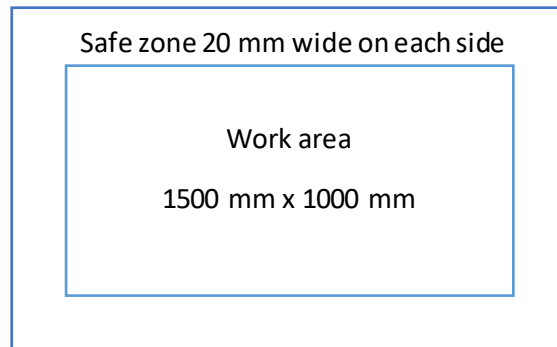
Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.



Evropský sociální fond
Praha a EU: Investujeme do vaší budoucnosti

Specification of the punch press device

The work area of the punch press is 1500 mm x 1000 mm. There is a 20mm wide border around the whole area called safe zone.



The punch press has a moving head which may move freely over the work area and the safe zone. Entering the safe zone with the center of the head is signaled by signals **SAFE_L**, **SAFE_R**, **SAFE_T**, **SAFE_B**. When the head is moved outside the work area and the safe zone, **FAIL** is signaled and the whole punch press stops. To recover from this condition, a reset is required.

The head is controlled by **PWM_X** and **PWM_Y**, which expects a [PWM](#) signal. Due to the punch press being simulated, your PWM signal is limited into the range of **1 kHz – 250 kHz**. The duty cycle of the PWM determines the power applied to the motor. The motor is driven by an [H-bridge](#). To invert the desired direction you need to indicate this in **DIR_X** and **DIR_Y** and invert the PWM signal.

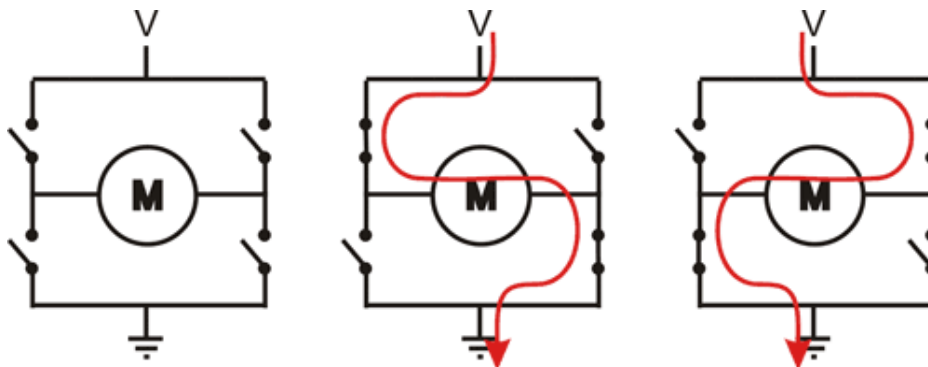


Figure: 1H-bridge

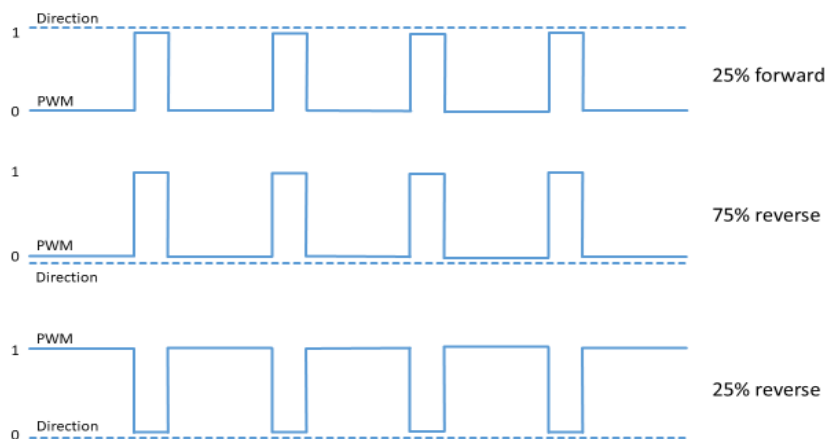


Figure 2: Result of deferent direction and PWM signals

The head is subject to friction forces, thus a certain minimal power is needed to get it into movement. When zero power is set, the head gradually slows down until it stops. It is however possible to apply opposite force to make the stop faster.

The movement of the head may be monitored using two [quadrature encoders](#) **ENC_X** and **ENC_Y**, which generate sequences 00, 01, 11, 10 for forward motion and sequences 10, 11, 01, 00 for backward motion. This sequence repeats once for 1 mm distance. You are required to use interrupts to catch encoder changes.

To punch a hole using the head, signal **PUNCH** is used. The signal has to remain set for at least 1ms. After that, it has to be cleared for at least 1ms. After the punch, it takes some time for the head to be lifted up its original position. This is signaled by **HEAD_UP**. When the signal is set, another punch may be performed or the head may be moved.

To perform the **PUNCH**, the head has to be still – that means its velocity along X as well as Y axis must be less than 0.0001 m/s (per axis). The head has to stay still until **HEAD_UP** is signaled. A failure to do so will result in **FAIL** mode. To recover from this condition, reset is required.

The signals are mapped as described below.

Output ports:

Pin	Name	Description
PB7	PWM_X	Power applied along the X axis.
PC0	DIR_X	Direction for the motor along the X axis.
PB6	PWM_Y	Power applied along the Y.
PC1	DIR_Y	Direction for the motor along the Y axis.
PC2	PUNCH	Signal to punch a hole at the current head position.

Input ports:

Pin	Name	Description
PC3	ENC_X	Quadrature encoder along the X axis.
PC4		
PC5	ENC_Y	Quadrature encoder along the Y axis.
PC6		
PC7	SAFE_L	Set when the center of the head is in the area left of the work area.
PC8	SAFE_R	Set when the center of the head is in the area right of the work area.
PC9	SAFE_T	Set when the center of the head is in the area top of the work area.
PC10	SAFE_B	Set when the center of the head is in the area bottom of the work area.
PC11	HEAD_UP	Set when the head is ready for move or punch.
PC12	FAIL	Signals the error condition. Reset is needed.

Execution of the platform

There will be provide two STM32F4 board connected together. The bottom one runs the punch press simulator and the upper one is dedicated to run your controller. Please connect the following in order to successfully run the simulation, its visualizer, and your code.

Connections

Board	Description
Top board MINI USB	Flashing your controller, connect to PC
Middle board UART	Your controller UART output, connect to USB serial in order to see the debugging output. This is internally connected to UART2 (GPIO_AF7_USART2, GPIO A pins 2 & 3)
Bottom board MICRO USB	Simulation output necessary to run visualizer, connect to PC

Run visualizer

```
java -jar /afs/ms/u/b/bures/ers-labs/visualizer.jar -h
```

The visualizer reads the file **punches.in**, which states positions of planned punches. The format of the file is the following:

```
<x coordinate of the 1st punch in mm from left>;<y coordinate of the 1st punch in mm from top>  
<x coordinate of the 2nd punch in mm from left>;<y coordinate of the 2nd punch in mm from top>  
<x coordinate of the 3rd punch in mm from left>;<y coordinate of the 3rd punch in mm from top>  
<x coordinate of the 4th punch in mm from left>;<y coordinate of the 4th punch in mm from top>  
...
```

The actual punches are produced in similar format to file **punches.out**.

For your convenience copy the following files into your directory:

```
/afs/ms/u/b/bures/ers-labs/punches.in  
/afs/ms/u/b/bures/ers-labs/visualizer.properties
```

The visualizer.properties file can be used to conveniently configure the visualizer.