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 some variant of a PID controller.

The application (called "controller") should be written for an STM32F4 board. The punch press simulator runs on another STM32F4 board providing output through serial port and feedback via dedicated pins that are connected to the controller. The serial port output can be viewed in a Java-based visualization application that is provided as a part of the assignment. The visualization is run on a laptop or desktop computer.

Report

The report should:

- Elaborate on the assignment and derive explicit real-time requirements (essentially replicating the design method taught at the course).
- Identify tasks and their scheduling parameters. Identify any synchronization.
- Explain how the respective tasks are mapped to activities in the implementation (e.g., interrupt handler, task scheduled in the main loop via timeline scheduling, FreeRTOS task).
- Describe and document measurements of the execution time of respective tasks. This should explain why the measured value can be considered as the worst case.
- Provide a systematic and detailed argument why all the tasks are guaranteed to meet their deadlines. This should rely on the theory on schedulability analysis taught in the lecures.

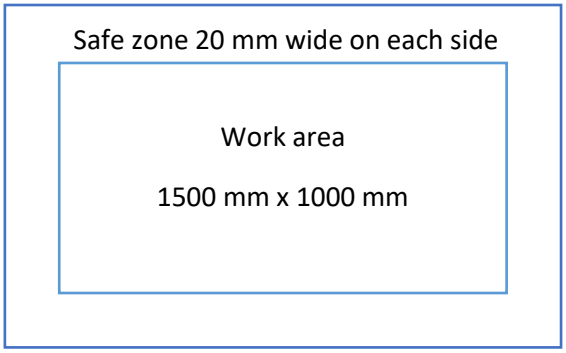
Submission

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

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, reset is required.

The head is controlled by **PWM_X** and **PWM_Y**, which expects a PWM signal. Since the punch press is simulated, your PWM signal should limited to the range of 100 kHz – 200 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.

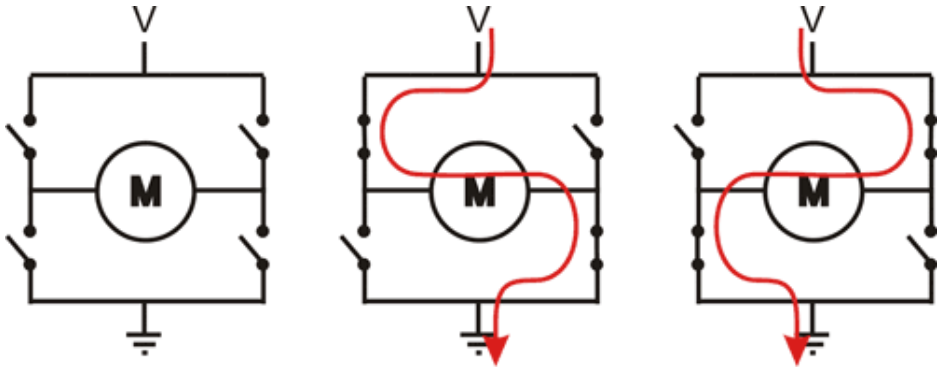


Figure: 1H-bridge

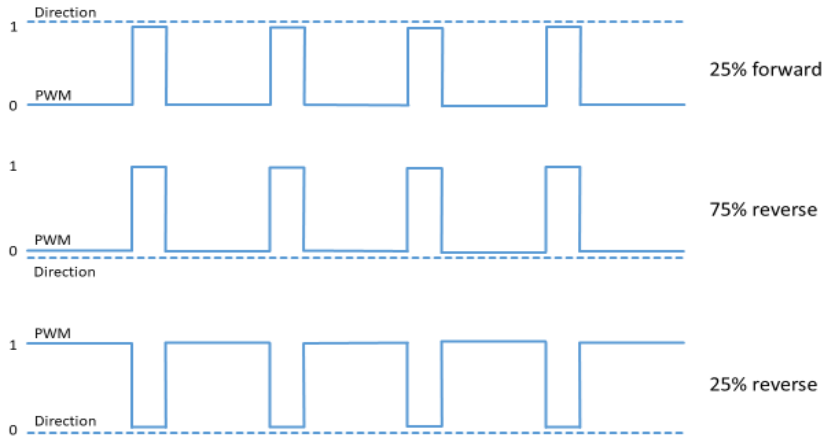


Figure 2: Result of deferent direction and PWM signals

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 it 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. The controller has to be able to handle the maximum speed of the head (i.e. when full duty cycle is actuated on the PWM).

To punch a hole using the head, signal **PUNCH** is used. The signal must remain set for at least 1ms. After that, it must be cleared for at least 1ms. After the punch, it takes some time for the head to be lifted 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 must be still – that means its velocity along X as well as Y axis must be less than 0.001 m/s (per axis). The head must 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.

Upon restart, the controller must wait at least 1ms to allow the simulator to properly initialize itself. Any signals values read before are not defined.

After reset/power up, the controller should be in stopped state. It should start the movement only after the blue (USER) button is pressed on the controller board.

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 **PC4**	ENC_X	Quadrature encoder along the X axis.
PC5 **PC6**	ENC_Y	Quadrature encoder along the Y axis.
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.

The simulation platform

You will be provided with two boards connected via a PCB. The board on the bottom side of the PCB runs the punch press simulator. The board on top of the PCB is dedicated to run your controller. Please connect the following to successfully run the simulation, its visualizer, and your code as indicated below.

Normally the simulator starts at a random position inside the work area. If PB1 on the simulator is grounded (by connecting it to the neighboring GND using a jumper), the simulator starts at a pre-defined position 100mm,100mm from the top-left corner of the work area. This is to just ease debugging. Your solution should work well with the random initial position.

Connections

Board	Description
Top board MINI USB	Flashing your controller, connect to PC
Middle board UART	Your controller UART output, connect to USB serial to see the debugging output. This is internally connected to UART2 (GPIO_AF7_USART2, GPIO A pins 2 & 3). Connect the USB-UART cable as follows: black – GND, green – RX, white – TX.
Bottom board MICRO USB	Simulation output necessary to run visualizer, connect to PC (it shall be detected as virtual serial port).

Visualizer

The state of the simulation can be visualized using the visualizer available along with this assignment. Unzip the visualizer and run the punchpressvis.bat (Windows) / punchpressvis (Linux) script stored under punchpressvis / bin.

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**.