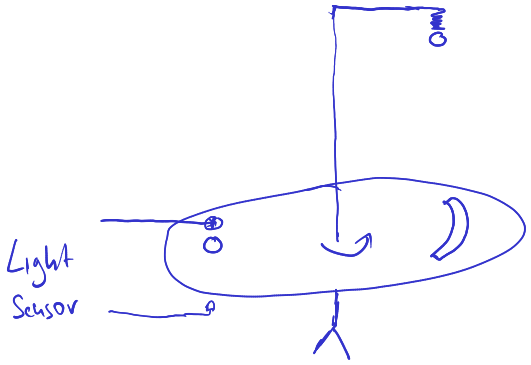
Exercise sheet 2 – Ball drop exercise

Goals:

- Use of FreeRTOS
- Understand basic timing calculations
- Implement different real-time programming techniques
- Understand and feel hard/firm real-time problems

Implementation of a hard/firm real-time system: the ball drop exercise



The idea of the ball drop exercise is to demonstrate the use case of hard/firm real-time. Here, a ball is held by a magnet. Below is a rotating disc with a hole. The ball must now be dropped at the right time so that it falls through the small hole in the rotating disc. If the timing is not correct, the ball falls onto the disc and the hard/firm real time is hurt. Programming this is very challenging and requires a guaranteed fast response time with mathematical calculations as part of the real-time programming.

Aspects of the exercise:

- Practising and experiencing hard/firm real-time
- Parallel and absolutely predictable programming methods
- Mathematical calculations as part of real-time programming
- Using sensors and actuators to control the hardware

Exercise 2.1: Determine and calculate the required timings (theoretical)

(a) Free fall time of the ball until the disc is reached. *Hint:* Use the formula $s = 0.5 \times g \times t^2$

- s: distance (fall height) of magnet to disc in m
- g: gravitational acceleration constant 9.80665 m/s^2
- t: free fall time in sec

(b) Free fall time of the ball until the ball is completely through the disc.

(c) Time the ball needs through the disc.

(d) The minimum possible rotation time in ms (≈ at max. velocity) of the disc. *Hint: The ball needs some time to fall through the small hole.* Consider a width of the hole of about 5.5 cm.

(e) Required rotation angle of disc between light sensor and ball drop hole.

(f) Determine the formula of how to calculate the wait time until the ball has to be dropped. Assume that you have already a variable called `rotation_time_ms` which contains the current rotation time in ms. *Hint: You may consider the circumference of the disc and the angles, as well as the time, the ball needs through the hole.*

Exercise 2.2: Implement and test the basic mechanisms

Hint: For debugging purposes, you can use the serial connection. For that, the `log_task()` task function is already implemented. To log something from a task, use the `log_message()` function, and from an ISR, use `log_message_ISR()`, respectively.

(a) Import the given project template into Sloeber from `sheet_02/erts_ex_2_ball_drop`.

(b) Follow the TODOs at the very beginning and modify `FreeRTOSConfig.h`.

(c) Implement the button press mechanism and control the state machine (variable `control_state`) within the `button_ISR()` ISR function. Use an appropriate software solution for the button debounce.

(d) Implement the ball drop mechanism (release magnet) in the `drop_ball_task()` task function and test it.

(e) Implement the rotate disk (sending a PWM signal) mechanism using the value of the potentiometer in the `motor_control_task()` task function. The `motor_control_task()` task function is triggered from a timer (timer 5). *Hint: You may use the `TimerFive` library and choose a 10 Hz rate.*

(f) Implement the hole detection in the `light_sensor_ISR()` ISR function.

Exercise 2.3: Implement the full ball drop example

(a) Connect all software parts and complete the ball drop example

- Depending on the `control_state` variable, control the disc rotation in `motor_control_task()`, if not already done.
- Inside the `light_sensor_ISR()`, determine the rotation speed in ms (save it to the `rotation_time_ms` variable). *Hint: You can use the `millis()` function from the Arduino framework.*
- Inside the `light_sensor_ISR()`, determine the wait time in ms until the ball should be dropped (release magnet) and save it to the `wait_time_drop_ball_ms` variable.
- Inside the `light_sensor_ISR()`: if `control_state == DROPPING`, start counting the disc turn and lift (`V()`) the `ball_drop_semaphore` after 2 turns.
- Inside the `drop_ball_task()`, block (`P()`) on the `ball_drop_semaphore`, until it is released. When it is released, wait for `wait_time_drop_ball_ms` and then drop (release magnet) the ball.

(b) Test the ball drop with varying rotation speeds.

Exercise 2.4: Improvements

(a) Instead of the software debounce in `button_ISR()`, use a hardware debounce solution.

Exercise 2.5: Questions

(a) Is it possible to use a software timer instead of the hardware timer to trigger the `motor_control_task()` task function?

(b) Determine the limitations of the system.

(c) Can the real-time requirements always be fulfilled?

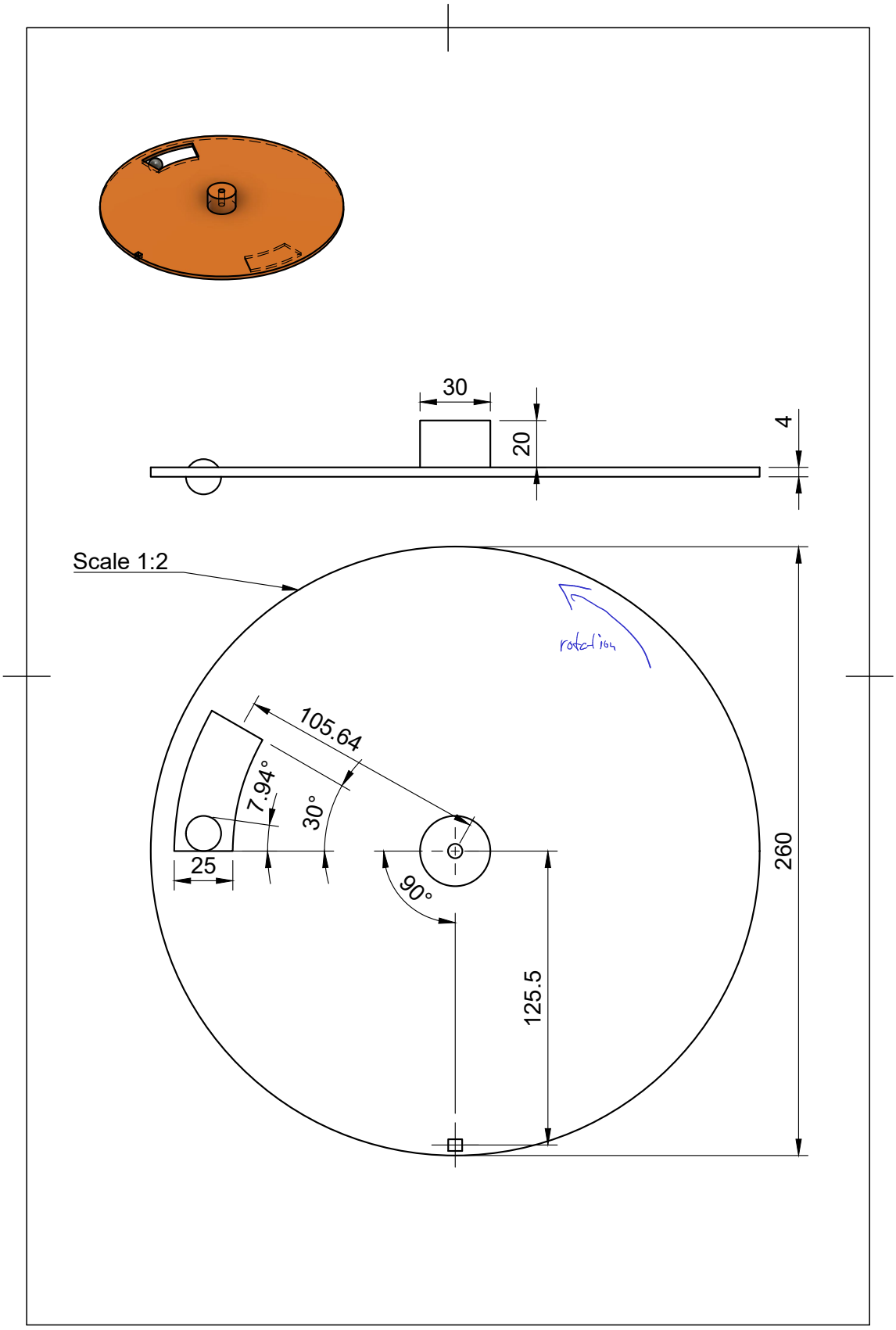
(d) Suggest improvements to the ball drop hardware and software.

Exercise 2.6: Measurements

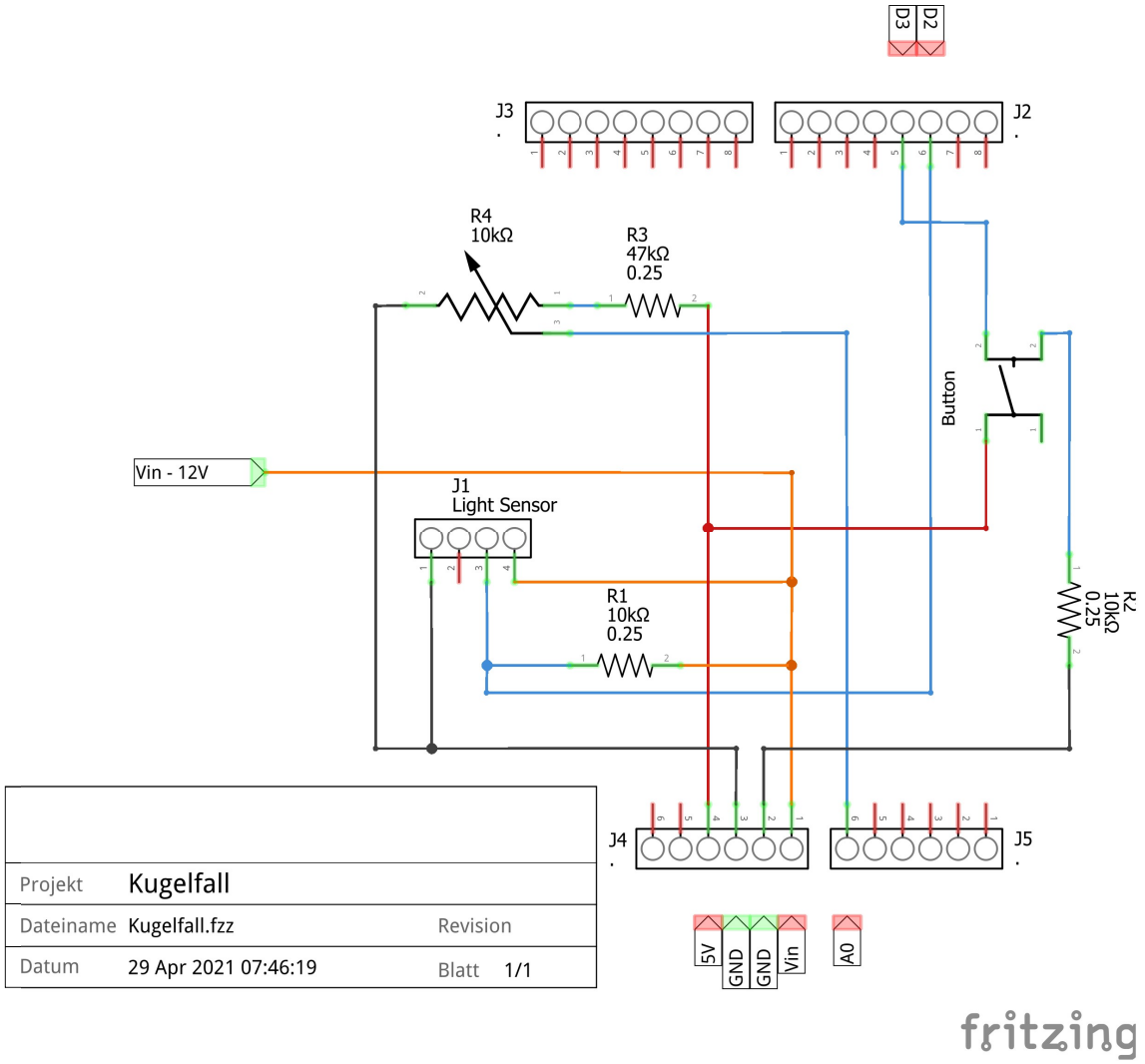
(a) Try to measure the button bounce problem.

(b) Try to measure the PWM signal with different rotation speeds.

Disc details:



Wiring


D3
D2
J3
J2
R4
10kΩ
R3
47kΩ
0.25
Button
Vin - 12V
J1
Light Sensor
R1
10kΩ
0.25
R2
10kΩ
0.25
J4
J5
5V
GND
GND
Vin
A0
Projekt Kugelfall
Dateiname Kugelfall.fzz
Revision
Datum 29 Apr 2021 07:46:19
Blatt 1/1
fritzing

Informal software architecture: