Notes for Inverted Pendulum

September 2022

1 Getting Started

The manual for the pendulum can be found in the Documents folder, moreover it contains information about the modeling of the pendulum and the datasheets of the components. The current version of the code is *Swing Up and Stabilization.ino* and the corresponding controller design is *simulation/acceleration_input*.

Please also observe:

- Remember to turn on the inverted pendulum always in the correct order (POWER ON $\rightarrow$ MCU $\rightarrow$ DRV $\rightarrow$ RESET).
- Make sure that the pendulum doesn't touch the table before powering it.
- Do not touch the rail or the cart while the pendulum is powered on via MCU/DRV, the cart is locked onto the rail at this point.

2 Remarks

The velocity of the inverted pendulum is controlled by the frequency of the PWM signal that drives the step input of stepper driver. It remains to be checked what the upper bound for the achievable velocity is.

Note that the pendulum cannot be controlled by the velocity as an input, therefore, we consider the acceleration the control input for designing and implementing the controller. This acceleration is then integrated to calculate the velocity before commanding it to the stepper.

2.1 PWM generation

In the program, the frequency is set using *ledCWriteTone* and has a upper limit of 78.125 kHz. The command for chaning the duty cycle of the signal is not used since only the frequency of the step signal is relevant.

The three parameters (channel, frequency, resolution) are required to attach the PWM channel to an actual GPIO of the ESP32. The ESP32 PWM controller has 16 independent channels that can be configured to generate PWM signals with different properties. The number of the channel can be chosen more or less arbitrarily, so we chose the first one, 0. You can keep it constant. The resolution parameter is irrelevant since it is only needed for PWM signals with different duty cycle. The function ledcWriteTone uses only 50 percent duty cycle, therefore, it is used only to setup the PWM output. The frequency is an essential parameter. Basically, it represents the velocity of the cart. This parameter is described in user manual of the Inverted Pendulum Device in section 2.4.3. **WARNING!!!** After setting up the communication interface on the pulse pin in Arduino code which controls the velocity of the cart via a PWM signal **(LINES IN THE CODE: ledcSetup(CHANNEL, FREQUENCY, RESOLUTION); ledcAttachPin(PULSE PIN, CHANNEL);)** do not use the **pinMode** command on the pulse pin. Using this command overwrites the **ledcSetup** command and the motors do not respond to **ledcWriteTone** command later.

2.2 Arduino Support Package

Matlab and Simulink provide support packages for Arduino compatible boards which include the ESP 32 on the inverted pendulum. However, the hardware encoders of the ESP32 for the incremental encoders are not supported. So, the usage of the support packages is not feasible.

2.3 MPC

There are two fundamental options for running MPC for the pendulum. The simplest is to have a PC connected to the pendulum run the MPC optimization. As a faster an alternative it might be desired to run the MPC directly on the microcontroller of the pendulum. This requires significantly more effort as toolbox for MPC is needed. Basically there are online MPC, which solves the MPC program online and explicit MPC, which has a parametrized version of the optimal control law.

For online MPC, you can use any software package or code generator for embedded convex optimization in C. Very popular are CVXGEN or qpOASES. Note that these are QP solvers, so they can only be used for linear MPC. A toolbox for nonlinear MPC, Acado and Acados are available.

For explicit MPC, the toolbox MPT3 can be used to generate an explicit form of MPC. Subsequently, the toolbox offers the option to generate the C-code of the controller. Using this option comes with two drawbacks. First, its memory footprint has to be small enough to upload on board. Second, the C-code has to be adjusted and transformed into libraries readable for Arduino IDE or any other IDE you are using. This action requires advanced programming experiences.

3 Useful Links

- OCL, the manufcatuturer of the pendulum
- MATLAB Support Package for Arduino
- System Modelling for Inverted Pendulum
- PWM Control for ESP32 Tutorial
- Timer Tutorials for ESP32 - 1 Timer Tutorials for ESP32 - 2
- MATLAB - Arduino Compatibility
- CVXGEN
- qpOASES
- MPT3