**WHEELCHAIR PROJECT**

Motor Control Libraries for STM32:

1. X-CUBE-MCSDK: <https://www.st.com/en/embedded-software/x-cube-mcsdk.html>

STM32100 SDK kit: <https://www.st.com/en/embedded-software/stsw-stm32100.html>

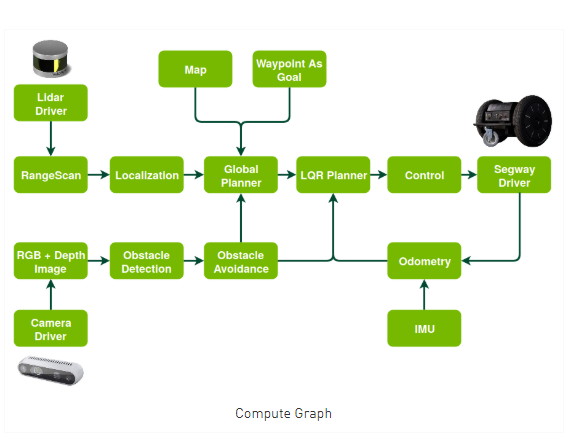
This library is used basically for 3 phase PMSM and BLDC motor with FOC control and firmware configuration. However, I am not sure whether it can be used for our wheelchair project since wheelchair contain Brushless DC Motors. Would like to discuss with you further.

1. X\_NUCLEO\_IHM04A1 Motor Control library: <https://os.mbed.com/teams/ST/code/X_NUCLEO_IHM04A1/>

Based on the last report, I had selected this motor controller and sent the block diagram on Thursday. This is the basic Mbed library based on L6206 motor control driver expanded with IHM04A1. **We can try to implement this library.**

Control block in Nvidia Robotic Platform-Issac SDK:

Issac SDK provides the tools and API for developing autonomous vehicles. Issac Robot engine contains the framework used for robotic applications. Below is the block diagram/compute blocks



For the wheelchair application, we will be basically focussing on the control and odometry blocks. The control block receives a motion “PLAN” from LQR planner and odometry section computes the linear and angular speeds of the vehicle.

The documentation for the ISSAC SDK can be found: <https://docs.nvidia.com/isaac/isaac/doc/index.html>

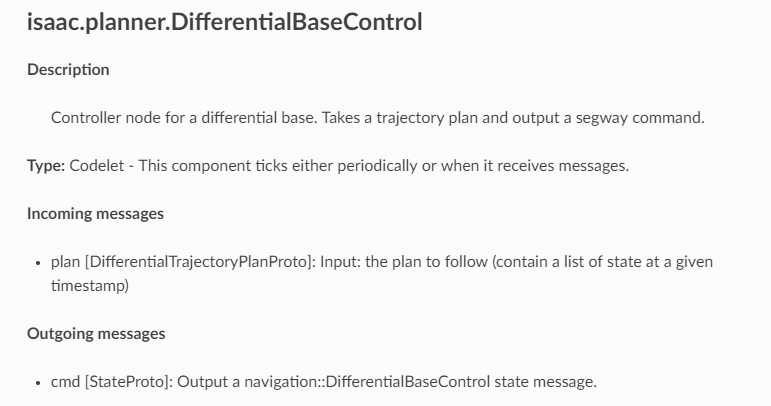
Message API: <https://docs.nvidia.com/isaac/isaac/doc/message_api.html>

Component API: <https://docs.nvidia.com/isaac/isaac/doc/component_api.html>

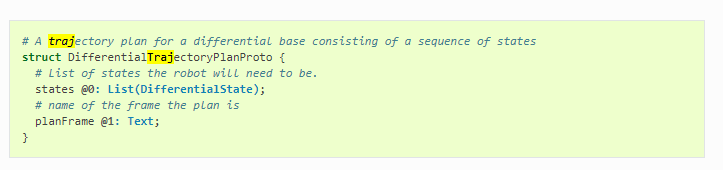
**CONTROL:**

As per the above component link, the API in the control section is implemented by planner section ( starts with issac.planner…). This codelet is implemented in the sample packages provided by Nvidia and can be used directly with Nvidia Jetson TX2.

1. Differential Base control



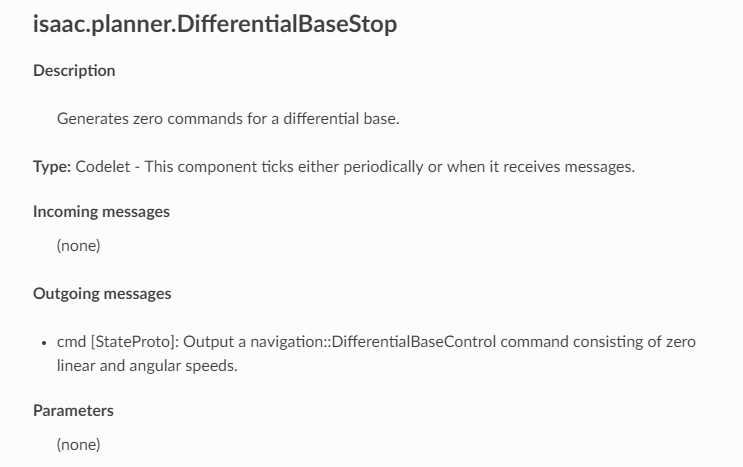
This receives a input of type “plan” from lqr planner (block diagram). It is a message of type struct “DifferentialTrajectoryPlanProto” which defines states and planframe. For wheelchair project, we can set a trajectory path in LQR planner and provide the dedicated path for wheelchair to move.



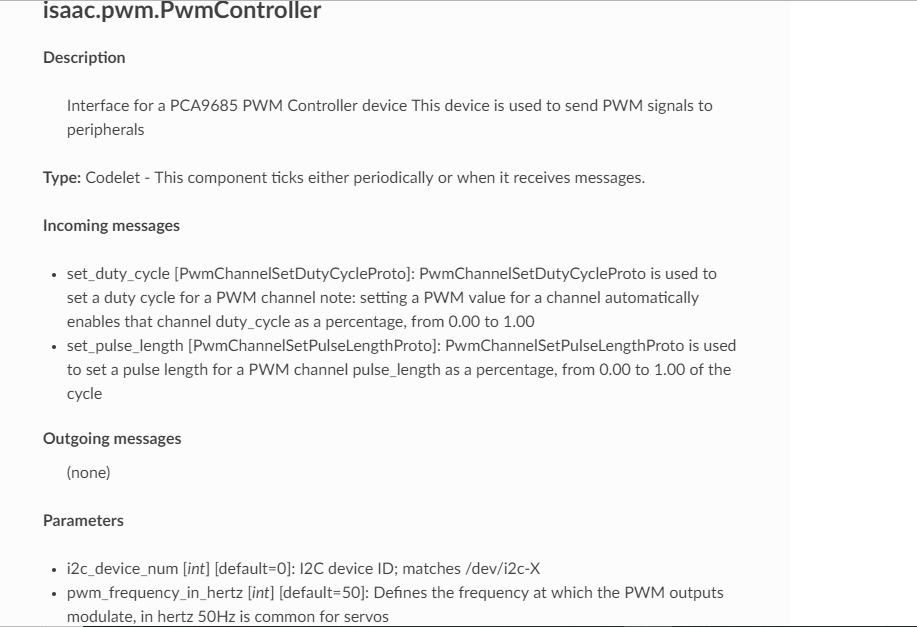
Output is cmd message of struct type “StateProto” which gives command directly to the driver, in this case could be the STM Discovery Kit.

1. Differential Base Stop

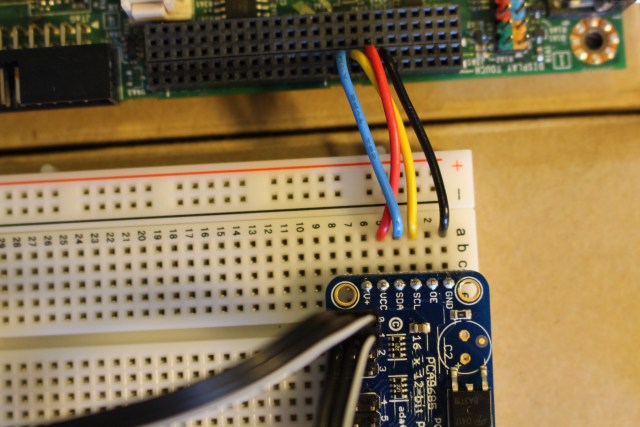
A direct structure API to stop the wheelchair. The linear and angular speeds will be set to zero.



1. PWM Controller



The PWM Controller API sends the PWM Duty Cycle and PWM length from PCA9685 PWM Controller(<https://www.adafruit.com/product/815>). PWM commands are sent from Nvidia Jetson to PCA9685 controller using I2C connections.



PCA9685 PWM controller

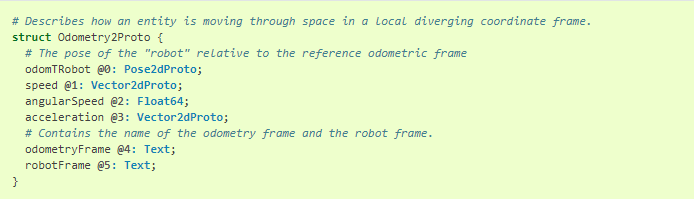
J21 connector of TX2

PCA9685 also offer an option for Arduino shield, which can be used for STM Discovery Kit(<https://www.electrodragon.com/product/pca9685-arduino-servo-shield/>)



**ODOMETERY:**

Integrates (2D) odometry for a differential base control to estimate it’s motion. It control the vehicle movements and message type is “Odometry2Proto”.



Sample code:

**namespace** isaac {

**namespace** navigation {

*// Integrates (2D) odometry for a differential base to estimate it's*

*// ego motion.*

**class** **DifferentialBaseOdometry** : **public** alice::Codelet {

**public**:

void start() **override**;

void tick() **override**;

*// Incoming current dynamic state of the differential base which is*

*// used to estimate it's ego motion in an odometry frame.*

ISAAC\_PROTO\_RX(DifferentialBaseStateProto, state)

*// Outgoing ego motion estimate for the differential base.*

ISAAC\_PROTO\_TX(Odometry2Proto, odometry)

*// Maximum acceleration to use (helps with noisy data or wrong data*

*// from simulation)*

ISAAC\_PARAM(double, max\_acceleration, 5.0)

*// The name of the source coordinate frame under which to publish*

*// the pose estimate.*

ISAAC\_PARAM(std::string, odometry\_frame, "odom")

*// The name of the target coordinate frame under which to publish*

*// the pose estimate.*

ISAAC\_PARAM(std::string, robot\_frame, "robot")

*// 1 sigma of noise used for prediction model in the following order:*

*// pos\_x, pos\_y, heading, speed, angular\_speed, acceleration*

ISAAC\_PARAM(Vector6d, prediction\_noise\_stddev, \

(MakeVector<double, 6>({0.05, 0.05, 0.35, 0.05, 1.00, 3.00})));

*// 1 sigma of noise used for observation model in the following order:*

*// speed, angular\_speed, acceleration*

ISAAC\_PARAM(Vector3d, observation\_noise\_stddev, \

(Vector3d{0.25, 0.45, 2.0}));

*// This is the pose under which the ego motion estimation will be*

*// written to the pose tree.*

ISAAC\_POSE2(odom, robot)

**private**:

...

};

} *// namespace navigation*

} *// namespace isaac*

ISAAC\_ALICE\_REGISTER\_CODELET(isaac::navigation::DifferentialBaseOdometry);

The ISSAC\_TX receives control message and ISSAC\_RX will send the odometry information.