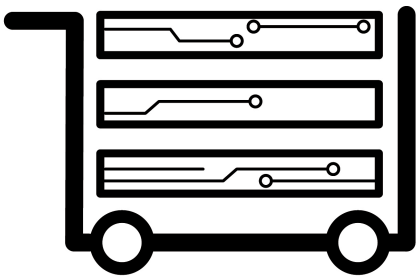
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
THE UNIVERSITY OF TEXAS AT ARLINGTON

DETAILED DESIGN SPECIFICATION
CSE 4317: SENIOR DESIGN II
SUMMER 2016



SMART CART

Autonomous Tool Cart

TEAM HAWT-WHEELS
SMART CART

BRIAN WONG
DAVID HARVEY
DENNIS OTIENO
PEGGY SOH
OR ZOARETS

Revision History

Revision	Date	Author(s)	Description
0.1	08.01.2016	DO, OZ	Document Creation
0.2	08.17.2016	PS, BW	Complete Draft

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

1.1 PRODUCT CONCEPT

The Smart Cart will provide assistance by being an autonomous carrier. It shall be able to do the following:

- Follow a "master" to the designated destination
- Avoid collision with other objects such as walls and people
- Include an integrated power supply
- Holonomic mobility

1.2 SCOPE

The main function of the Smart Cart is to help its users carry tools from one point to another. It shall have an integrated power supply to allow it charge or power tools. Unique features of the Smart Cart is that it shall identify and follow "master" by using the Intel RealSense and possess a holonomic mobility. These shall allow Smart Cart to easily maneuver and avoid collisions with other objects. Smart Cart shall require external input from the user, user shall wear a colored band that is recognized and used by Smart Cart to follow the "master". Smart Cart shall produce outputs inform of messages to let its user know whether or not it has successfully identified its "master".

1.3 KEY REQUIREMENTS

The Smart Cart shall have four layers that work in unison to allow the cart to autonomously navigate and follow a specified user. The four layers in the system are:

- Smart Cart layer
- Power Supply layer
- Imaging and Navigation layer
- Crab Drive layer

The four layers will be discussed more in the following sections

2 SYSTEM OVERVIEW

The four layers of Smart Cart shall interface as follows. The Power supply shall interface with the cart at both the kill and the manual mode switches. It shall interface with the Imaging and navigation layer at the camera interface and with the crab drive layer at the drive system interface. The crab dive shall interface with the cat at the wheels and the imaging and navigation layer at the camera interface.

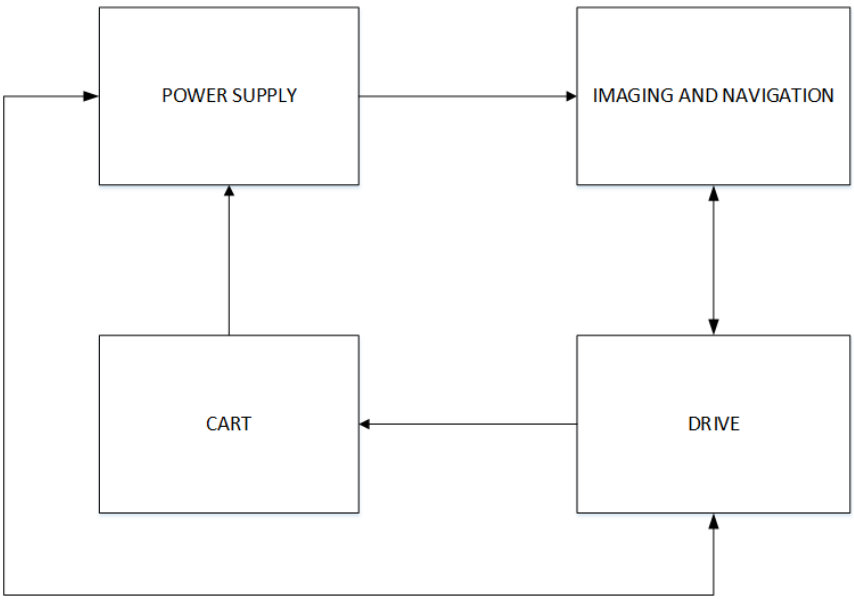


Figure 1: System architecture

2.1 POWER SUPPLY LAYER

Shall include a 12 volt deep cycle battery, a voltage regulator, a voltage converter and a recharge switch. The main function of this layer shall be to provide power to the rest of the layers and also to provide power for charging and powering on tools

2.2 CART LAYER

Shall include hardware components such as the cart, wheels, a kill switch and a manual mode switch. This shall be the cart to be driven by the crab drive, powered by the power supply and guided by the imaging and navigation layers

2.3 CRAB DRIVE LAYER

Shall include the motor control system, the drive system interface and the chain system. This component shall be responsible for driving the cart from one point to another

2.4 IMAGING LAYER

Shall include mainly the camera and the camera interface and shall be responsible for guiding the cart towards it's master and away from obstacles.

3 POWER SUPPLY SUBSYSTEM

The power supply layer is one of the most important layers because it provides stable power supply to the cart. The parts in this subsystem are responsible for powering every other subsystem and any external tools or equipment.

3.1 DEEP CYCLE BATTERY

A rechargeable battery that will provide consistent power up to 12 V.

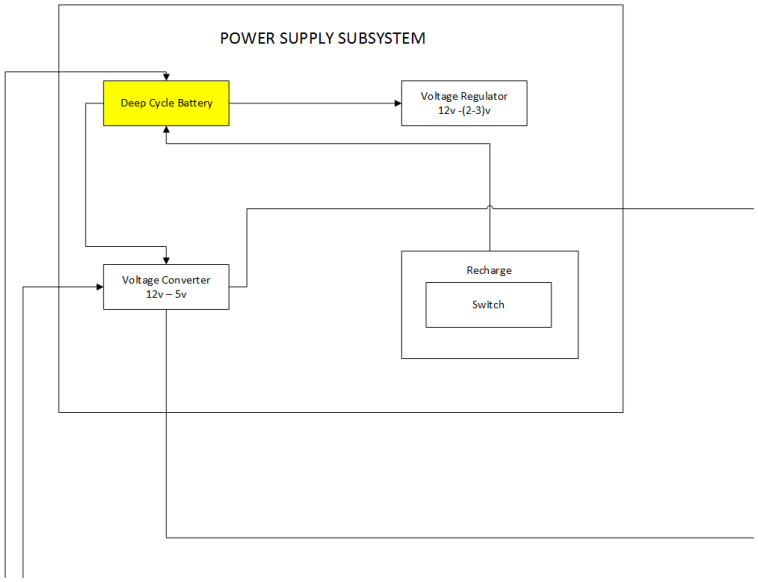


Figure 2: Deep Cycle Battery Subsystem description diagram

3.1.1 ASSUMPTIONS

- The battery is charged.
- The battery is properly connected to the cart
- The battery is fully functional
- The battery is connected to the cart at all times

3.1.2 RESPONSIBILITIES

The deep cycle battery subsystem responsibilities are as follows:

- Provide stable power supply to the other subsystems after being converted from 12v to 5v by the voltage converter.
- Enable stable power to external tools and equipment after being converted from 12V to 2 - 3V through the voltage regulator.

3.1.3 SUBSYSTEM INTERFACES

Table 2: Deep cycle battery subsystem interfaces

ID	Description	Inputs	Outputs
N/A	Power integrated power supply	N/A	Voltage regulator
N/A	Power electronic components	N/A	Voltage converter
N/A	Safety killswitch	Killswitch	N/A
N/A	Safety manual mode	Manual-mode switch	N/A

3.2 VOLTAGE REGULATOR

The voltage regulator is designed to convert the 12V voltage from the deep cycle battery to 2 - 3V and it will mainly be used to power up external tools and equipment.

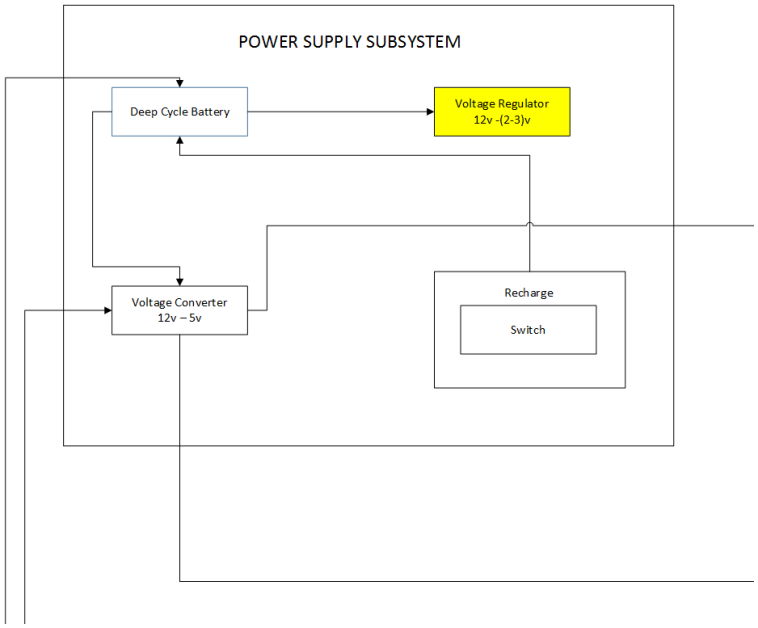


Figure 3: Voltage Regulator Subsystem description diagram

3.2.1 ASSUMPTIONS

Assumptions made are as follows:

- The voltage from the battery will be 12v.
- The voltage required fro external tools and equipment will be between 2-3v.

3.2.2 RESPONSIBILITIES

- To enable and regulate constant voltage level for powering external tools and supplies.

3.2.3 SUBSYSTEM INTERFACES

Table 3: Voltage Regulator Subsystem Interfaces

ID	Description	Inputs	Outputs
N/A	Receive power	Deep Cycle Battery	N/A

3.3 RECHARGE

The Recharge subsystem will be used to to ensure that Smart Cart will be charged without having too much electricity going to the battery or other cart components

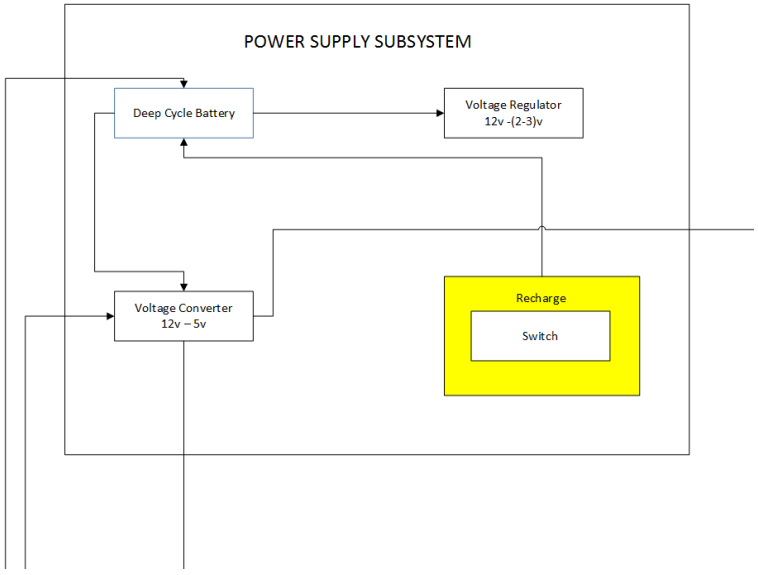


Figure 4: Recharge Subsystem

3.3.1 ASSUMPTIONS

Assumptions are as follows:

- The Recharge subsystem will ensure the safety of the tools and equipment.

3.3.2 RESPONSIBILITIES

The responsibilities of the Recharge subsystem are as follows:

- A switch will disable the battery from the powering the Smart Cart components while it is being charged

3.3.3 SUBSYSTEM INTERFACES

Table 4: Recharge Subsystem Interfaces

ID	Description	Inputs	Outputs
N/A	Switch o safety	Switch	Deep cycle battery

3.4 VOLTAGE CONVERTER

The voltage regulatlor is designed to convert the 12V from the deep cycle battery to a 2-3V to be used to power up smart cart equipment

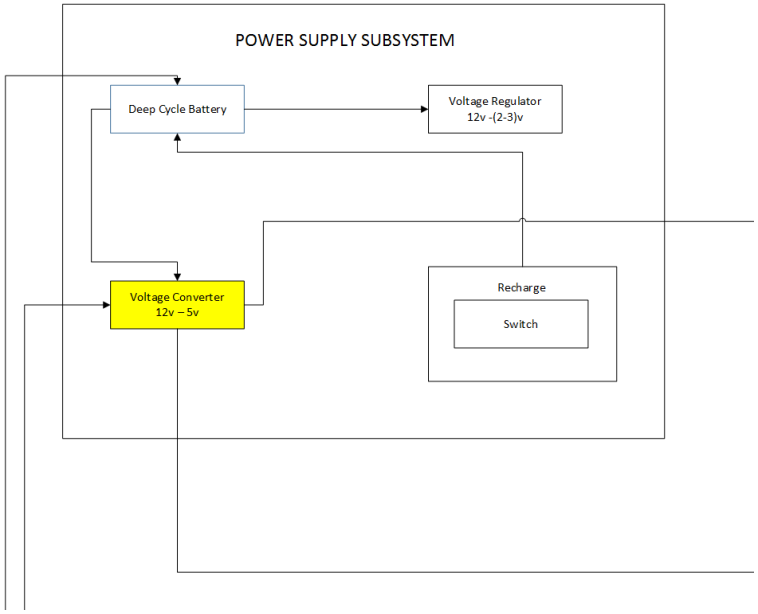


Figure 5: Voltage Converter Subsystem

3.4.1 ASSUMPTIONS

Assumptions made are as follows:

- The voltage from the battery will be 12V.
- The voltage required by the cart's components will be 5V

3.4.2 RESPONSIBILITIES

The voltage converter subsystem responsibilities are as follows:

- Enable stable power to the other subsystems, which include the crab drive system and image processing after being converted from 12V to 5V through the voltage converter.

3.4.3 SUBSYSTEM INTERFACES

Table 5: Voltage Converter Subsystem Interfaces

ID	Description	Inputs	Outputs
N/A	Receive Power	Deep Cycle battery	N/A
N/A	Safety Manual Mode	Manual-mode switch	Crab-drive system Imaging and Navigation

4 IMAGING AND NAVIGATION SUBSYSTEM

The imaging and navigation layer consists of the Intel RealSense Camera, ultrasonic sensors and imaging software to produce navigation data to be sent to the drive system interface. The navigation data that will direct the crab drive system to follow a user as well as obstacle avoidance

4.1 SENSORS SUBSYSTEM

Sensors include the Intel RealSense camera and ultrasonic sensors. The RealSense will identify a user to follow and the ultrasonic sensors will be used for obstacle avoidance. The sensors will provide the necessary information to the camera interface needed to create navigation data.

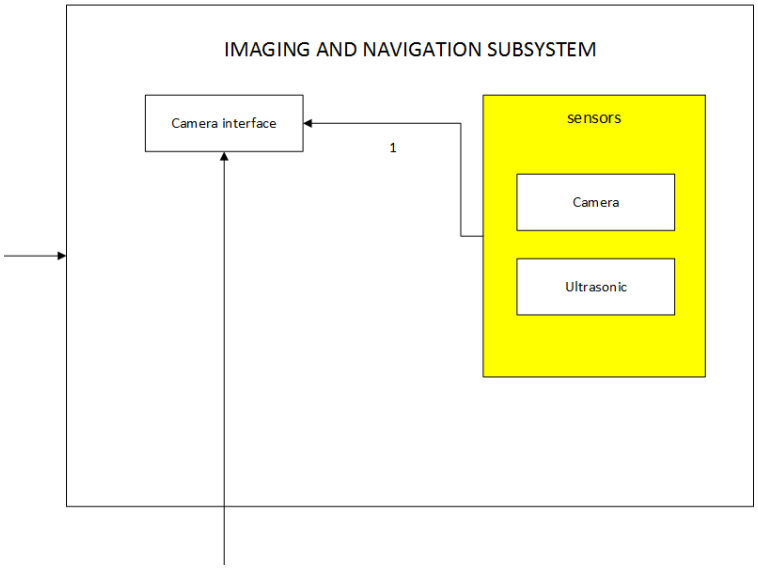


Figure 6: Sensors subsystem description diagram

4.2 ASSUMPTIONS

Assumptions made are as follows:

- The user will be wearing a colored wristband that identifies them as the master
- The user will stay in front of the Intel RealSense to avoid possible errors, such as the cart loosing vision of its master

4.3 RESPONSIBILITIES

The sensor subsystem responsibilities are as follows:

- Process image and ultrasonic data in real time
- keep the designated master within the image frame
- Gather accurate data to send to the camera interface

4.4 SUBSYSTEM INTERFACES

Table 6: Sensors subsystem interface

ID	Description	Inputs	Outputs
1	Send sensor data	Camera Ultrasonic	Camera interface

4.5 CAMERA INTERFACE SUBSYSTEM

The camera interface is responsible for receiving data from sensors and calculating the required navigation data and sending it to the crab drive's system interface. The navigation data will include require speed, direction needed and distance from obstacles.

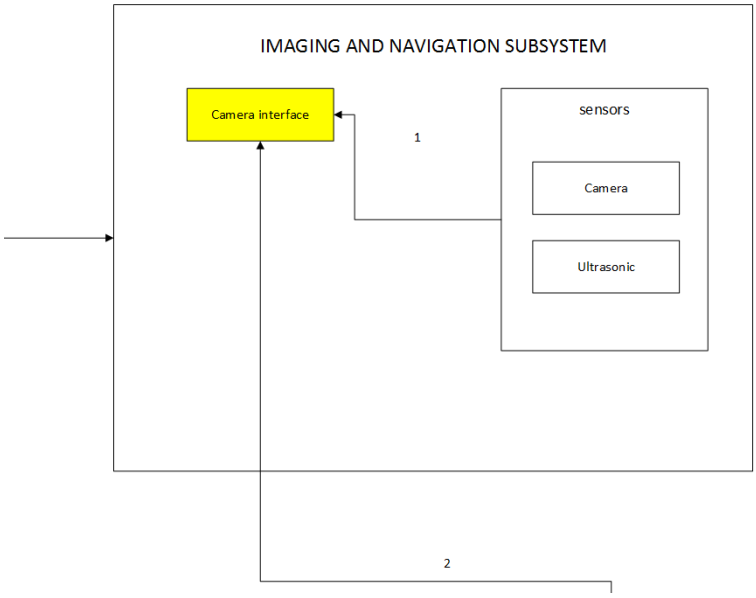


Figure 7: Camera Interface subsystem description diagram

4.6 ASSUMPTIONS

Assumptions are as follows:

- Accurate data is being sent from the sensors
- The camera interface will be able to communicate with the drive system interface

4.7 RESPONSIBILITIES

The camera subsystem responsibility are as below:

- Calculate and process the navigation data
- Communicate necessary information to the drive system interface

4.8 SUBSYSTEM INTERFACES

Table 7: Camera Interface Subsystem interfaces

ID	Description	Inputs	Outputs
2	Calculating navigation data	Sensors	Drive System interface

5 CRAB DRIVE SUBSYSTEM

The crab drive layer is a vital layer to this project. It's major function shall be to make Smart Cart mobile. It comprises comprises such vital elements as the micro-controller, the motor-controllers and the drive system interface.

5.1 MOTOR CONTROL SUBSYSTEM

The motor control subsystem will contain a roller chain that will be responsible for steering the the cart and maintain a consistent orientation for all the wheels, a central motor to coordinate the movement of the four wheels and wheel motors to drive the wheels

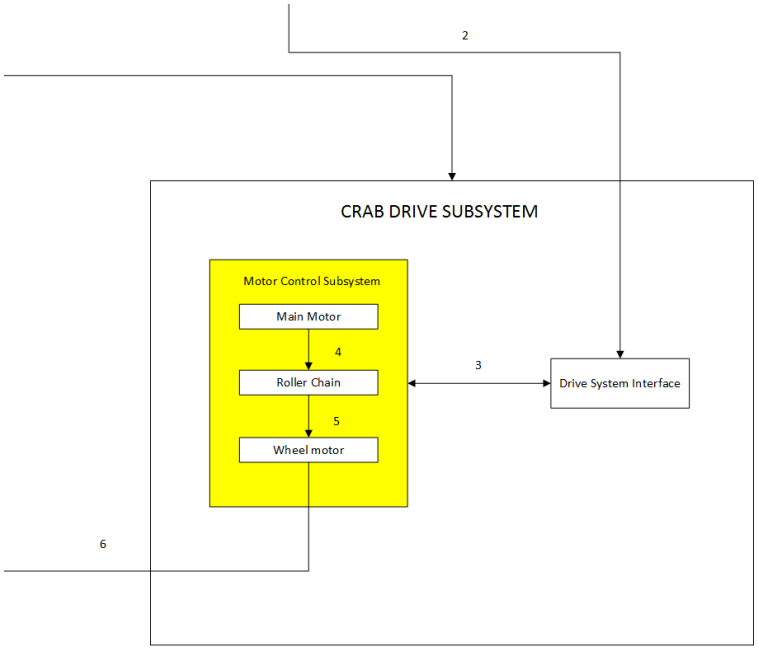


Figure 8: Motor Control subsystem description diagram

5.2 ASSUMPTIONS

Assumptions made are as follows:

- The main motor will correctly steer the wheels with the roller chain
- The main motor will have a method to accurately keep wheel orientation data
- The wheel motors will move the wheels accordingly

5.3 RESPONSIBILITIES

The main motor subsystem's responsibilities are as follows:

- Receive steering instructions from the drive system interface and accurately perform instructions.

5.4 SUBSYSTEM INTERFACES

Table 8: Motor Control subsystem interface

ID	Description	Inputs	Outputs
3	Feedback data	N/A	Drive system interface
6	Steer Wheels	Drive system interface	Wheel motors

5.5 DRIVE SYSTEM INTERFACE SUBSYSTEM

The drive system interface s designed to communicate with the navigation subsystem's camera interface it will get navigation data from the camera interface to steer the main drive motor as well as receiving feedback information for the wheel orientations.

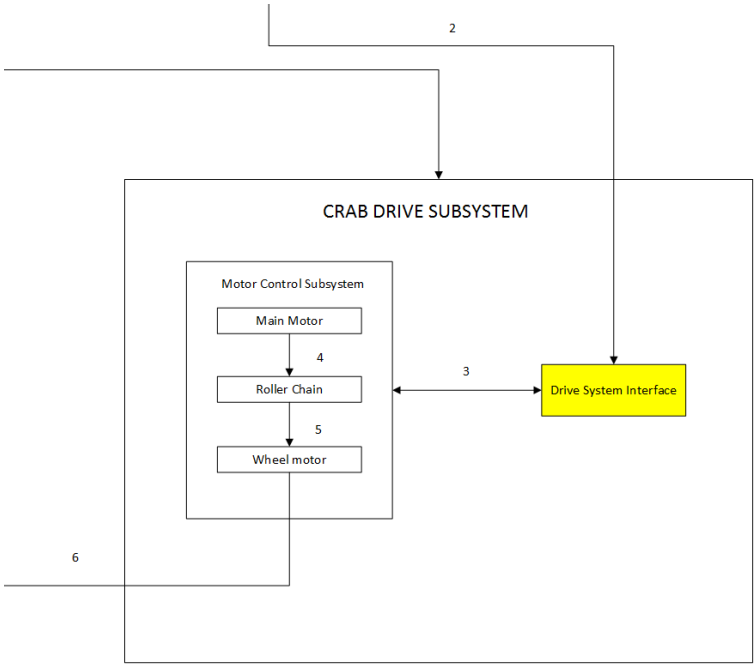


Figure 9: Drive system interface subsystem description diagram

5.6 ASSUMPTIONS

Assumptions made are as follows:

- The drive system interface will receive accurate navigation data from the camera interface.
- The drive system will receive accurate feedback from the motor control subsystem.
- The drive system interface will receive accurate feedback data from the main motor about wheel orientation

5.6.1 RESPONSIBILITIES

The responsibilites for drive system interface are as follows:

- Send navigation data to the motor control
- Receive wheel feedback data

5.6.2 DRIVE SYSTEM INTERFACE SUBSYSTEM INTERFACES

Table 9: Drive System Interface Subsystem Interface

ID	Description	Inputs	Outputs
2	Receive navigation data	Camera interface	Motor Control
3	Receive wheel data	Motor control	Camera interface

5.7 TEENSY MICRO-CONTROLLER

A Teensy 3.2 micro-controller shall be used to control and coordinate the wheels via the motor-controllers.

5.7.1 SUBSYSTEM PROGRAMMING LANGUAGES

The Teensy micro-controller shall be programmed in C++

5.7.2 SUBSYSTEM DATA STRUCTURES

Data will be transmitted from a PC via USB to the Teensy.

5.8 MOTOR CONTROLLERS

The Smart Cart utilizes a Teensy Micro-controller in order to control the motorized wheels.

5.8.1 SUBSYSTEM PROGRAMMING LANGUAGES

The motor controller shall be programmed in C++ and shall receive directional data from the micro-controller.

6 CART SUBSYSTEM

The cart subsystem is largely the cart itself. it shall have pace for carrying user tools together with a charging area for user to put charging equipment, kill switch, manual mode switch and the wheels.

6.1 MANUAL MODE SWITCH SUBSYSTEM

The manual mode switch subsystem will be used to switch the cart to manual mode by disconnecting the power supply to the crab drive

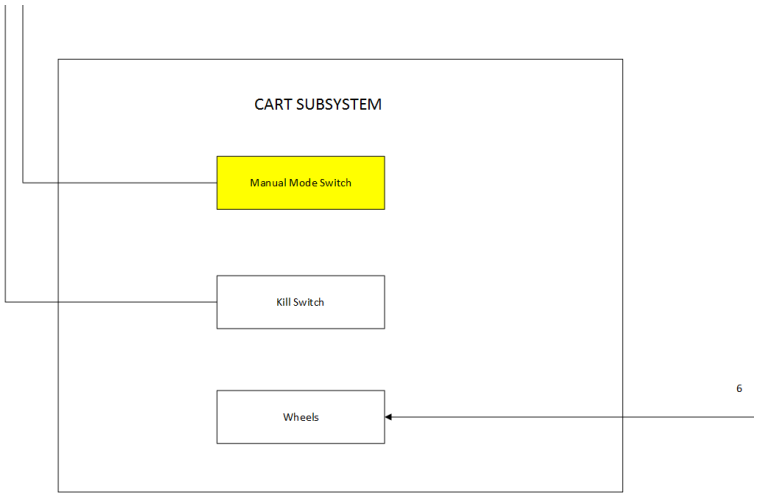


Figure 10: Manual Switch subsystem description diagram

6.1.1 ASSUMPTIONS

Assumptions made are as follows:

- The manual mode switch subsystem will ensure the safety of users and around the user
- The manual mode switch subsystem will not power off Smart Cart, but simply disable autonomous motion

6.1.2 RESPONSIBILITY

The manual mode switch subsystem responsibilities are as follows:

- The switch will be used to disable the autonomous motion of the cart.
- The switch will ensure safety of the users and the surrounding people

6.1.3 MANUAL MODE SUBSYSTEM INTERFACE

Table 10: Manual mode switch subsystem interface

ID	Description	Inputs	Outputs
N/A	Disable converter	N/A	Voltage converter

6.2 KILLSWITCH

The killswitch subsystem will be used to ensure that the Smart Cart will be immediately turned off.

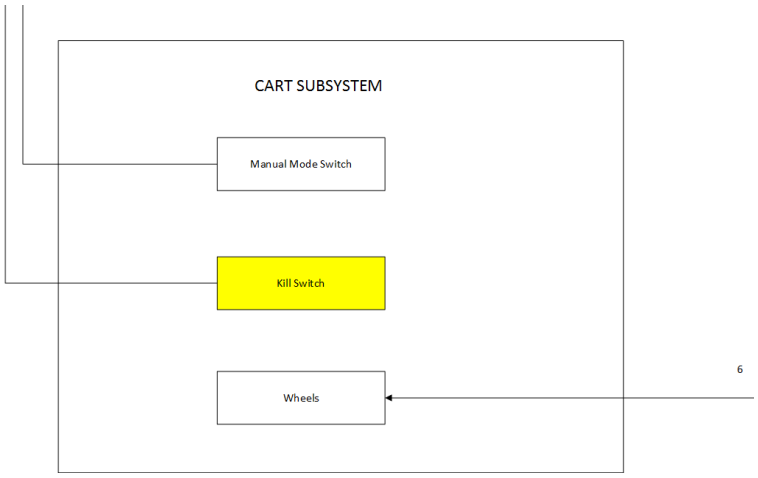


Figure 11: Kill Switch subsystem description diagram

6.2.1 ASSUMPTIONS

Assumptions made are as follows:

- Power to Smart Cart components will be disabled
- Power to external tools and equipment will be disabled

6.2.2 RESPONSIBILITIES

The killswitch subsystem responsibility are as follows:

- the switch will disable the battery from powering Smart Cart and its components

6.2.3 KILLSWITCH SUBSYSTEM INTERFACE

Table 11: Sensors subsystem interface

ID	Description	Inputs	Outputs
N/A	Disable battery	N/A	Deep cycle battery

6.3 WHEEL SUBSYSTEM

The wheels subsystem will be used to provide movement to the Smart Cart. The wheels will be controlled by the Crab Drive layer.

6.3.1 ASSUMPTIONS

Assumptions made are as follows:

- The wheels will be able to turn in the direction sort
- The wheels will move harmoniously

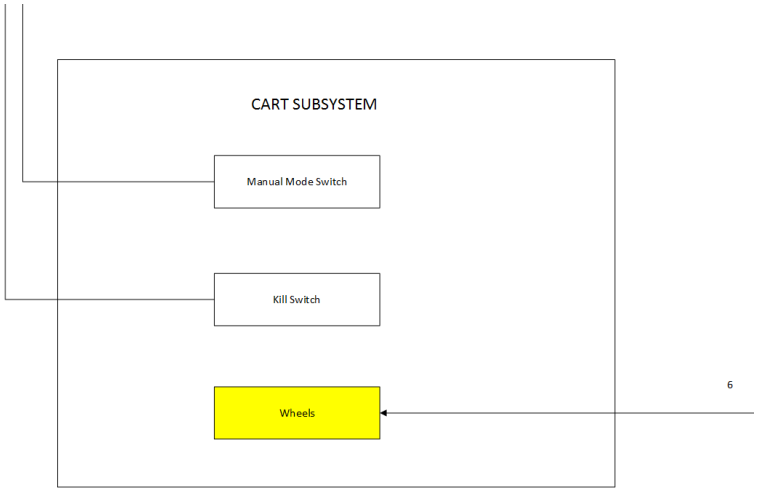


Figure 12: Wheels subsystem description diagram

6.3.2 RESPONSIBILITIES

The wheel subsystem responsibilities will be as follows

- Turn and move the cart from one point to anther
- Take turns as directed by the wheel motors

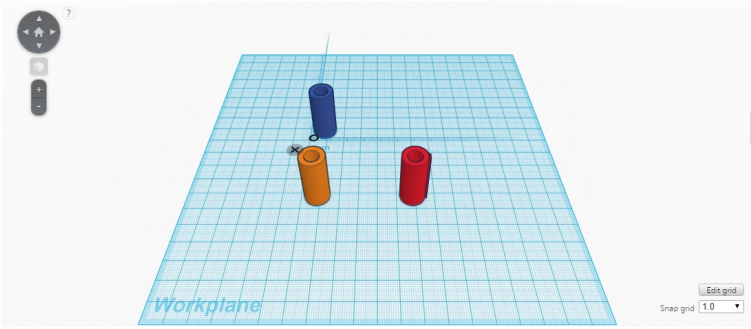
6.3.3 WHEEL SUBSYSTEM INTERFACES

Table 12: Sensors subsystem interface

ID	Description	Inputs	Outputs
5	Move Smart Cart	Wheel Motors	N/A

7 Appendix A

7.1 CAD Design for Central Motor Sprocket



7.2 Joystick Input Test Code

```
//A0(15) = green, A1(16) = yellow, A2(17) = orange, A3(18) = red
#define downPin 15
#define upPin 16
#define rightPin 17
#define leftPin 18
int X = 0;
int Y = 0;
int PrevX = X;
int PrevY = Y;

void setup() {
  Serial.begin(9600);

  //Joystick setup
  pinMode(downPin, INPUT);
  digitalWrite(downPin, HIGH);
  pinMode(upPin, INPUT);
  digitalWrite(upPin, HIGH);
  pinMode(rightPin, INPUT);
  digitalWrite(rightPin, HIGH);
  pinMode(leftPin, INPUT);
  digitalWrite(leftPin, HIGH);
}

void loop() {
  joystick( &X, &Y );
  if (X != 0 && (X != PrevX || Y != PrevY)){
    result(X,Y);
  }
  else if (Y != 0 && (X != PrevX || Y != PrevY)) {
    result(X,Y);
  }
```

```
  delay(300);
}

// [-1, 0, 1] map to [left/down, none, right/top]
void joystick(int* X, int* Y){

  //HIGH = non-trigger
  if( !digitalRead( leftPin ) ){
    *X = -1;
  }else if( !digitalRead( rightPin ) ){
    *X = 1;
  }else{
    *X = 0;
  }

  if( !digitalRead( upPin ) ){
    *Y = 1;
  }else if( !digitalRead( downPin ) ){
    *Y = -1;
  }else{
    *Y = 0;
  }

}

void result(int X, int Y ) {
    if (X == 1){
      Serial.print("Right\t");
    }
    else if (X == -1) {
      Serial.print("Left\t");
    }
    PrevX = X;
    if (Y == 1){
      Serial.print("Up\n");
    }
    else if (Y == -1) {
      Serial.print("Down\n");
    }
    else{
      Serial.print("\n");
    }
    PrevY = Y;
}
```

7.3 Motor Driver Test Code

```
/* LED Blink, Teensyduino Tutorial #1
   http://www.pjrc.com/teensy/tutorial.html
```

```
   This example code is in the public domain.
*/

// Teensy 2.0 has the LED on pin 11
// Teensy++ 2.0 has the LED on pin 6
// Teensy 3.x / Teensy LC have the LED on pin 13

//A0(15) = green, A1(16) = yellow, A2(17) = orange, A3(18) = red
const int LED_PIN = 13;
const int PWM_PIN = 14;
const int DIR_PIN = 10;
const int HB_PWM = 3;
const int HB_INA = 4;
const int HB_INB = 5;
const int DOWN_PIN = 15;
const int UP_PIN = 16;
const int RIGHT_PIN = 17;
const int LEFT_PIN = 18;
int X = 0;
int Y = 0;
int PrevX = X;
int PrevY = Y;

// the setup() method runs once, when the sketch starts
void setup() {
  // initialize the digital pin as an output.
   Serial.begin(9600);

  pinMode(LED_PIN, OUTPUT);
  pinMode(PWM_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);
  pinMode(HB_PWM, OUTPUT);
  pinMode(HB_INA, OUTPUT);
  pinMode(HB_INB, OUTPUT);
  pinMode(DOWN_PIN, INPUT);
  pinMode(UP_PIN, INPUT);
  pinMode(RIGHT_PIN, INPUT);
  pinMode(LEFT_PIN, INPUT);

  for(int i = 0; i < 25; i++)
  {
    digitalWrite(LED_PIN, HIGH);
    delay(50);
    digitalWrite(LED_PIN, LOW);
    delay(50);
  }
```

```
  digitalWrite(PWM_PIN, LOW);
  digitalWrite(DOWN_PIN, HIGH);
  digitalWrite(UP_PIN, HIGH);
  digitalWrite(RIGHT_PIN, HIGH);
  digitalWrite(LEFT_PIN, HIGH);
}

// the loop() methor runs over and over again,
// as long as the board has power
// pwm low = off
// 0 forward
void loop() {
  joystick( &X, &Y );
  if (X != 0 && (X != PrevX || Y != PrevY)){
    result(X,Y);
  }
  else if (Y != 0 && (X != PrevX || Y != PrevY)) {
    result(X,Y);
  }
  delay(300);
}

void joystick(int* X, int* Y){
  //HIGH = non-trigger
  if( !digitalRead( LEFT_PIN ) ){
    *X = -1;
    turnLeft();
  }else if( !digitalRead( RIGHT_PIN ) ){
    *X = 1;
    turnRight();
  }else{
    *X = 0;
  }

  if( !digitalRead( UP_PIN ) ){
    *Y = 1;
    moveForward();
  }else if( !digitalRead( DOWN_PIN ) ){
    *Y = -1;
    moveBackward();
  }else{
    *Y = 0;
  }
}

void result(int X, int Y ) {
    if (X == 1){
      Serial.print("Right\t");
```

```
    }
    else if (X == -1) {
      Serial.print("Left\t");
    }
    PrevX = X;
    if (Y == 1){
      Serial.print("Up\n");
    }
    else if (Y == -1) {
      Serial.print("Down\n");
    }
    else{
      Serial.print("\n");
    }
    PrevY = Y;
}

void turnLeft() {
  digitalWrite(HB_PWM, HIGH);
  digitalWrite(HB_INA, LOW);
  digitalWrite(HB_INB, HIGH);
  delay(1350);
  digitalWrite(HB_PWM, LOW);
}

void turnRight() {
  digitalWrite(HB_PWM, HIGH);
  digitalWrite(HB_INB, LOW);
  digitalWrite(HB_INA, HIGH);
  delay(1350);
  digitalWrite(HB_PWM, LOW);
}

void moveForward() {
  digitalWrite(PWM_PIN, HIGH);
  digitalWrite(DIR_PIN, HIGH);
  delay(1000);
  digitalWrite(PWM_PIN, LOW);
}

void moveBackward() {
  digitalWrite(PWM_PIN, HIGH);
  digitalWrite(DIR_PIN, LOW);
  delay(1000);
  digitalWrite(PWM_PIN, LOW);
}
```

7.4 Final Demo Code

```
const int LED_PIN = 13;
const int PWM_PIN = 10;
const int DIR_PIN = 14;
const int HB_PWM = 3;
const int HB_INA = 4;
const int HB_INB = 5;
const int DOWN_PIN = 15;
const int UP_PIN = 16;
const int RIGHT_PIN = 17;
const int LEFT_PIN = 18;

int X = 0;
int Y = 0;
int PrevX = X;
int PrevY = Y;

void setup() {
  Serial.begin(9600);

  pinMode(LED_PIN, OUTPUT);
  pinMode(PWM_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);
  pinMode(HB_PWM, OUTPUT);
  pinMode(HB_INA, OUTPUT);
  pinMode(HB_INB, OUTPUT);
  pinMode(DOWN_PIN, INPUT);
  pinMode(UP_PIN, INPUT);
  pinMode(RIGHT_PIN, INPUT);
  pinMode(LEFT_PIN, INPUT);

  for(int i = 0; i < 25; i++)
  {
    digitalWrite(LED_PIN, HIGH);
    delay(50);
    digitalWrite(LED_PIN, LOW);
    delay(50);
  }

  analogWrite(HB_PWM, LOW);
  digitalWrite(PWM_PIN, LOW);
  digitalWrite(DOWN_PIN, HIGH);
  digitalWrite(UP_PIN, HIGH);
  digitalWrite(RIGHT_PIN, HIGH);
  digitalWrite(LEFT_PIN, HIGH);
}
```

```
void loop() {
  joystick( &X, &Y );
  if (X == 0  && Y == 0){
    stopMoving();
  }
  if (X != 0 && Y != 0){
    result(X,Y);
  }
 /* stopMoving();
  delay(3000);
  moveForward();
  delay(2000);
  stopMoving();
  delay(3000);
  moveBackward();
  delay(2000);*/
}

void joystick(int* X, int* Y){
  if( !digitalRead( UP_PIN ) ){
      *Y = 1;
      moveForward();
  } else if( !digitalRead( DOWN_PIN ) ){
      *Y = -1;
      moveBackward();
  } else{
      *Y = 0;
  }

  if( !digitalRead( LEFT_PIN ) ){
    *X = -1;
    if (*Y == -1){
      turnLeft();
    } else {
        turnRight();
    }
  } else if( !digitalRead( RIGHT_PIN ) ){
    *X = 1;
    if (*Y == -1){
      turnRight();
    }
    else {
      turnLeft();
    }
  } else{
    *X = 0;
  }
}
```

```
void result(int X, int Y ) {
    Serial.print( "X = ");
    Serial.print(X);
    Serial.print(" Y = ");
    Serial.print(Y);
    Serial.print('\n');

    PrevX = X;
    PrevY = Y;
}

void turnLeft(){
  digitalWrite(HB_PWM, HIGH);
  digitalWrite(HB_INA, LOW);
  digitalWrite(HB_INB, HIGH);
}

void turnRight(){
  digitalWrite(HB_PWM, HIGH);
  digitalWrite(HB_INB, LOW);
  digitalWrite(HB_INA, HIGH);
}

void moveForward(){
  analogWrite(PWM_PIN, 80);
  digitalWrite(DIR_PIN, HIGH);
}

void moveBackward(){
  analogWrite(PWM_PIN, 80);
  digitalWrite(DIR_PIN, LOW);
}

void stopMoving(){
 analogWrite(PWM_PIN, LOW);
 digitalWrite(HB_PWM, LOW);
 digitalWrite(HB_INA, LOW);
 digitalWrite(HB_INB, LOW);
}
```