OttoKart Operating Procedures

HARDWARE

Hardware Startup

- Obtain Key & LIDAR Box from Zane
- Mount LIDAR to OttoKart
    - Attach USB micro cable to LIDAR
    - Hand-tighten the mounting bolt to the front hitch.
- Electrical Startup
    - Insert Key and Turn to On Position
        - Do not use cart lights. This will drain the battery.
    - Turn both driver-side emergency switches clockwise until extended
    - Attach laptop to USB cable (See Software Startup for COM port settings)
    - Start Processing sketch and ensure connections
    - Engage manual control by pressing 'G' key on keyboard
- Backup Procedures
    - Remove the plastic cover from the steering control and store it in the OttoKart
    - Have a group member hold open dome flaps
    - Put OttoKart into reverse moving the forward/reverse switch by your legs until you hear the backup beeper
    - Use your feet to control the brake and accelerator. Use the manual control (green controller) to steer
        - Avoid engaging the steering while the OttoKart is stopped. Turning the wheels while stopped puts unnecessary strain on the motor
    - Slowly reverse the OttoKart out of the dome. Be careful not to snag the overhead camera and warning light on the dome
    - Once you are clear of the dome, put the OttoKart into forward using the forward reverse switch
    - Again, using your feet for braking and acceleration, and the manual control for steering, slowly drive the OttoKart to the testing area
- Remember to disengage manual control by pressing 'G' on the keyboard
    - You can always override the system by manually working the brake and accelerator pedals

Hardware Shutdown

- Return the OttoKart to the dome area
- Have a group member hold open the flaps while you slowly drive into the dome
    - Make sure the OttoKart is well-centered in the dome to make it easy for the next group to remove it
- Park the OttoKart in the center of the dome. Do not engage the parking brake.

- Quit the Processing program by pressing 'Q' on the keyboard
- Press both emergency switches until they lock down
- Turn the key to the off position and return it to the LIDAR box
- Cover the steering controls with the plastic cover
- Gently unplug the USB micro cable from the LIDAR and remove the mounting bolt
- Return the LIDAR to the LIDAR box and return to Zane

SOFTWARE

Software Startup

- Install the SLABtoUART driver and Arduino driver located here:
    - https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers
    - https://drive.google.com/open?id=127wizRVDi2d17JmV7fWpetEcmxhGc_ss
- Ensure that you have opened the OttoKart software in Processing 2.2 (other versions are not compatible with the libraries used)
- Plug in the USB cable to your computer
- Run the Processing sketch. On startup it will attempt to connect to the various devices. It will likely fail the first time because you must configure the COM ports for your computer.
    - COM ports are how your computer talks to the devices in the OttoKart such as the Arduino, camera, GPS, and LIDAR. Each computer will recognize these differently, so it is important to configure your computer to ensure it can connect. Even with the same computer this configuration can change from time to time, so it is important to recognize when a connection is not being made.
        - ***Camera Configuration*** - The first bit of information displayed in the Processing debug window is the Camera configuration. It might look something like this (PC/MAC results may vary):
            - [0] "name=USB 2.0 Camera #3,size=1280x720,fps=30"
            - [1] "name=USB 2.0 Camera #3,size=1280x720,fps=15"
            - [2] "name=USB 2.0 Camera #3,size=1280x720,fps=1"
            - [3] "name=USB 2.0 Camera #3,size=640x360,fps=30"
            - [4] "name=USB 2.0 Camera #3,size=640x360,fps=15"
            - [5] "name=USB 2.0 Camera #3,size=640x360,fps=1"
            - In this case, we want to configure the camera for 640x360 resolution and 30 frames per second. We can see this is number 3. So on the Settings tab in the software we will change the variable 'whichCam' to be equal to 3.
        - ***LIDAR Configuration*** - The next bit of information displayed will be the LIDAR connection. This should not require any changes to the Settings tab. You may occasionally get a 'timeout' error. If this

occurs, check all USB connections to ensure proper fit, as this is the most likely cause.

- ***Arduino and GPS Configuration*** - The Arduino and GPS COM ports may be difficult to recognize at first. The output will look something like this (PCs will look different)
    - [0] "/dev/cu.Bluetooth-Incoming-Port"
    - [1] "/dev/cu.SLAB_USBtoUART"
    - [2] "/dev/cu.usbmodem1412401"
    - [3] "/dev/cu.usbserial-141230"
    - [4] "/dev/cu.wchusbserial141230"
    - [5] "/dev/tty.Bluetooth-Incoming-Port"
    - [6] "/dev/tty.SLAB_USBtoUART" ← LIDAR
    - [7] "/dev/tty.usbmodem1412401" ← GPS
    - [8] "/dev/tty.usbserial-141230"
    - [9] "/dev/tty.wchusbserial141230" ← Arduino
    - To determine which port is which device, open the Serial Monitor in Arduino, pick a COM port, and set the baud rate to 9600.
        - If you have selected the GPS, the output in the Serial Monitor will look like this:
            - $GPVTG,,,,,,,,N*30
            - $GPGGA,,,,,,0,00,99.99,,,,,,*48
        - If you have selected the Arduino, the output in the Serial Monitor will look like this:
            - System Ready!
    - Once you have identified the correct COM ports, adjust the values for whichArduino and whichGPS to match the appropriate numbers on the settings tab.

Software Shutdown

- To quit the Processing sketch, press 'Q' on the keyboard
- Be sure to save any changes you've made to the Processing sketch (such as updated COM port settings).
- It can be good to save old versions of your software as a backup. That way if you make a mistake you can't undo, you can always go back to an older, known good version of the software.

Basic Software Operations

- ***User Interface & Algorithms***
    - Each team will compose their interface and algorithm code on the "Algorithm" tab. There is a master function for each of these challenges, i.e. interface(), avoid(), local(), and global(). You can write additional functions on this page, but they should all be called from within the appropriate master functions.

        - More details about the specific challenges can be referenced in the OttoKart Challenge Document
    - These functions are toggled on/off using keyboard commands. This will allow you to manually navigate to a testing area, e.g. VikingTrail, and when everything is set, run your algorithm to see how it performs. The keyboard commands for algorithms are as follows:
        - avoid() - Toggles by pressing the '1' key
        - local() - Toggles by pressing the '2' key
        - global() - Toggles by pressing the '3' key
        - interface() - Toggles by pressing the '4' key
- **Sensor Input**
    - *LIDAR*
        - LIDAR data is stored in an ArrayList of PVectors. This structure is called *lidarContacts*. A PVector has three floating point values stored as an X, Y, and Z. The values stored here are as follows:
            - X - X Position of contact point
            - Y - Y Position of contact point
            - Z - Distance of contact point from origin in millimeters
        - The origin point of the LIDAR data is located at point x = 200, y = 200
            - *Example*: A point located at x = 200, y = 100, would be located in front of the golf cart.
        - A common way to loop through all points in the LIDAR ArrayList looks like this:

```
for (int i = 0; i < lidarContacts.size(); i++){
        PVector p = lidarContacts.get(i);
        float x = p.x;          // X Coord of Contact
        float y = p.y;          // Y Coord of Contact
        float d = p.z           // Z Coord of Contact
}
```

    - *Camera*
        - The raw video feed for the camera is captured at 640 x 320 pixel resolution at 30 frames per second. It is stored as a PImage called *cam*.
        - The raw feed is automatically reduced to an 80 x 80 pixel PImage called *reducedVideo*. This reduced image is much easier and quicker to perform image processing calculations on.
        - Any operations performed should be stored in the 80 x 80 pixel PImage called *enhancedVideo*.
        - A common way to perform operations on PImages looks like this:

```
reducedVideo.loadPixels();              // Loads pixels from reducedVideo
enhancedVideo.loadPixels();             // Load pixels from enhancedVideo
for (int i = 0; i < reducedVideo.width * reducedVideo.height; i++){
```

```
        color c = reducedVideo.pixels[i];      // Gets a pixel from reducedVideo
        float r = c >> 16 & 0xFF;              // Gets the red value of the pixel
        float g = c >> 8 & 0xFF;               // Gets the green value of the pixel
        float b = c  & 0xFF;                   // Gets the blue value of the pixel

        color n = color(b, g, r);              // Example: swaps red and blue
values
        enhancedVideo.pixels[i] = n;           // Puts the pixel in enhancedVideo
}
enhancedVideo.updatePixels();                  // Refreshes pixels in enhancedVideo
```

- Screen Recording
  - The software has a built in feature to capture the screen while the software is running. This should be used to document testing and trial runs. To use it, do the following:
    - Press V to Start/Stop Screen Capture. A red circle in the upper left hand part of the screen will appear while recording.
    - To convert your capture file to video, do the following:
      - The capture file is a folder of images inside the Processing sketch folder (Go to Sketch->Show Sketch Folder to locate). Use the Movie Maker Tool (Go to Tools->Movie Maker) to create a .mov file from the images. Select "Same Size as Originals" from Options

- GPS
  - The GPS Sensor can sometimes take several minutes to lock onto enough satellites to target your position. A green circle will appear in the GPS panel when it is tracking. A red circle means it is not tracking yet.
  - GPS points are stored as PVectors. They store information in the following structure:
    - PVector.x = latitude
    - PVector.y = longitude
  - There are several ArrayLists of PVectors that you can use. They include:
    - GPSCoords - ArrayList of Historical GPS Locations
    - waypointStraight - Straight Trail Navigation
    - waypointCurved - Curved Trail Navigation
    - waypointSingle - Single Point Navigation
    - waypointMultiple - Multi Point GPS Navigation
  - There are also some included functions that will help you do simple operations with latitude and longitude points:
    - float bearing(PVector p1, PVector p2) - Takes two points and returns a floating point value that represents the bearing

between those Points, p1 (Current Coordinate) and p2 (Previous coordinate)
        - float distance(PVector p1, PVector p2) - Returns the distance (in meters) between two Points, p1 and p2
        - convertGPS(PVector p) - Returns a PVector with converted lat/lon coordinates into X/Y coordinates
- **Control Output**
    - Manual vs. Auto
        - Manual mode allows the OttoKart to be driven with the hardware controller (circuit board with buttons). Automatic gives control over to the Processing Sketch.
        - There is a variable that stores that control mode:
            - statusManual
                - true - Manual On
                - false - Manual Off
    - Steering
        - *Output* - You can send commands to turn the wheels of the OttoKart:
            - sendTurnLeft() // Left Turn - Begins turning wheel left
            - sendTurnRight() // Right Turn - Begins turning wheel right
            - sendTurnStop() // Stop Turn - Stops turning the wheel
        - Input - There is a simple Hall Effect Sensor that can help you understand how much the vehicle has turned.
            - steerCounter - This variable stores how much the wheels have turned between -7 and 7. (Negative numbers mean turning left, positive numbers mean turning right).
            - If the OttoKart tries to turn beyond these limits it will stop the turning to prevent damage to the vehicle.
                - This limit is called SafeSteer. If you accidentally lock the vehicle, you can return the vehicle to normal operation by disabling SafeSteer (Press '+'). Putting the OttoKart in Manual Control (Press 'g'), and manually adjusting the wheels back to the straight position. After doing so, press '=' to recenter and zero out the sensor.
        - There is a variable that stores the current status of the Steering
            - statusTurn
                - 0 - Stop Turn
                - 1 - Left Turn
                - 2 - Right Turn
        - *You should avoid turning too much while the OttoKart is stopped.*
        - *You should also avoid turning beyond the limits of the OttoKart. These actions can cause undue strain on the vehicle or break it.*
    - Acceleration

        - The Accelerator has 4 commands. Each moves the pedal to a preset position to set a certain speed (or lack thereof). Most challenges should use the Slow speed unless otherwise indicated.
        - sendAccelStop() // Accel Stop - Moves Accelerator to 0% Position
        - sendAccelSlow() // Accel Slow - Moves Accelerator to 25% Position
        - sendAccelWalk() // Accel Walk - Moves Accelerator to 50% Position
        - sendAccelRelease() // Accel Release - Moves Accelerator to 0% Position
        - There is a variable that stores the current status of the Accelerator
            - statusAccel
                - 0 - Accel Stop
                - 1 - Accel Slow
                - 2 - Accel Walk
                - 3 - Accel Release
    - Braking
        - Braking is similar to the Accelerator. Each moves the pedal to a preset position. You should make a note that sendBrakeStop *does not stop the kart* It stops braking...meaning the brake pedal will be released.
        - sendBrakeStop() // Brake Stop - Moves Brake to 0% Position
        - sendBrakeSlow() // Brake Slow - Moves Brake to 25% Position
        - sendBrakeWalk() // Brake Walk - Moves Brake to 50% Position
        - sendBrakeRelease() // Brake Release - Continually Moves Brake to 0% Position
        - There is a variable that stores the current status of the brake:
            - statusBrake
                - 0 - Brake Stop
                - 1 - Brake Slow
                - 2 - Brake Walk
                - 3 - Brake Release
                - 4 - Brake Full
- **Keyboard Commands**
    - System Controls
        - Q - Quits the Program
        - 1 - Toggles the Avoid Function
        - 2 - Toggles the Local Function
        - 3 - Toggles the Global Function
        - 4 - Toggles the Global Function
    - GPS Controls
        - W - Clears Historical Positions
        - E - Show No Position
        - R - Show Current Position
        - T - Show Historical Positions

        - Y - Show No Waypoints
        - U - Show Straight Path Waypoints
        - I - Show Curved Path Waypoints
        - O - Show Single Point Waypoints
        - P - Show Multipoint Waypoints
    - LIDAR Controls
        - None
    - Output Controls
        - ‘ - Full Release (Accel/Brake)
        - . - Full Stop (Accel/Brake/Steer)
        - Z - Left Turn
        - X - Right Turn
        - C - Stop Turn
        - = - Zero Turn Sensor
        - + - Toggle Safe Steering
        - V - Accel Stop
        - B - Accel Slow
        - N - Accel Walk
        - M - Accel Release
        - L - Brake Stop
        - K - Brake Slow
        - J - Brake Walk
        - H - Brake Release
        - ; - Brake Full
        - G - Toggle Manual Control
    - Camera & Video Recording
        - A - Start/Stop Screen Capture
        - S - Toggle Unprocessed Image vs. Processed Image
        - D - Toggle Color Picker
- **Code Reference**
    - Additional code examples and references are located on the Ref

WEEKLY MAINTENANCE

Battery Care *(Performed Weekly)*

- Attach the OttoKart to the golf cart charger and leave overnight
    - Check charge indicator on OttoKart to ensure full charge
- Attach each of the 12v batteries to car battery charger and leave overnight
    - Check voltmeters to ensure charge reaches ~12.3 volts or higher

Other

- Wipe down all surfaces with a damp cloth or baby wipes
- Ensure all wiring is secure and not frayed