Dronee: Project Writeup

ENSC 351 Froshees

Project GitHub Repository: SatireSage/Dronee: Beaglebone and Arduino Based Controller/Drone Unit

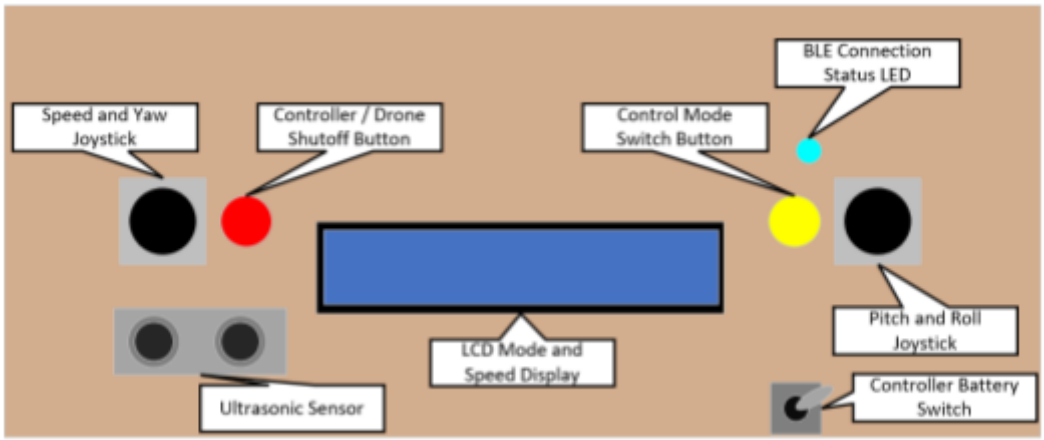
Authors: Sahaj Singh, Bryce Leung, Sukha Lee, Andrew Speers

Drone and Controller

Table of Contents:

System Explanation

Our system is a remote controlled drone, consisting of an Arduino Nano 33 IoT and a BeagleBone Green. The BeagleBone in this case is used as the controller interface for the user, while the Arduino is used as the drone that receives and interprets the commands sent.



BeagleBone Controller Layout

Dronee: Project Writeup

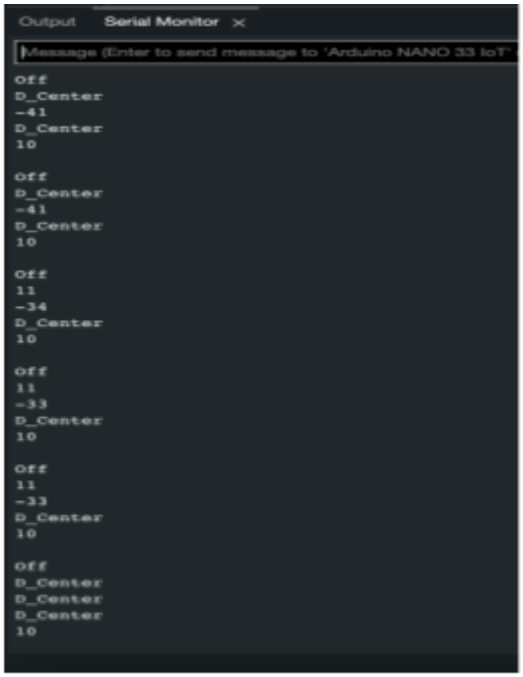
ENSC 351 Froshees

To start the controller and drone, the controller's battery switch must be flipped on to begin the supply of power to the controller and the drone must also be plugged into power. When power is supplied to the BeagleBone, the program begins to run through the utilization of Systemd. This removes the need for the user to manually have the beaglebone run the program as it will open the program upon startup. With the program running, the necessary threads and initializations start to allow the utilization of watchdog, joysticks, buttons, LCD display, ultrasonic sensor, LED, and the UART BLE module.

With the necessary initializations and threads running, the controller connects the drone through the BLE module which is used to send encoded packets from the controller to the drone. To have this work, the drone was set as the central bluetooth device while the controller was set as the peripheral bluetooth device. This was done as the UART BLE module is unable to read/write to specific BLE characteristics. To show the presence of a successful connection, an LED on the controller lights up when a successful connection has been made and turns off when a loss of connection is detected. The watchdog in this case, is used in a thread to check the connection status. When the disconnection flag has been activated, the watchdog is no longer hit resulting in timer running out. This leads to the controller rebooting to try to obtain a connection with the drone once again.

Startup Message From Controller

To control the drone, the program takes readings from the joysticks, ultrasonic sensor, and yellow button to be sent as encoded packets to the drone. The left joystick is used to manipulate the PWM values of the drone. By moving the joystick up and down, it increases or decreases the PWM value linearly at a rate between -10 and 10. The yaw is determined by moving the joystick the left and right, affecting the associated motors linearly between -28 (counterclockwise) to 28 (clockwise) PWM. The right joystick is used to control the pitch of the drone, by moving the joystick up and down. Changing the front and back motors of the drone linearly between -42 (backwards) to 42 (forwards) PWM. The right and left movements of the joystick controls the roll of the drone. Changing the left and right motors of the drone linearly between -28 (leftwards) to 28 (rightwards) PWM. This scheme forms the default control mode of the drone.



Arduino Printing Decode Packets

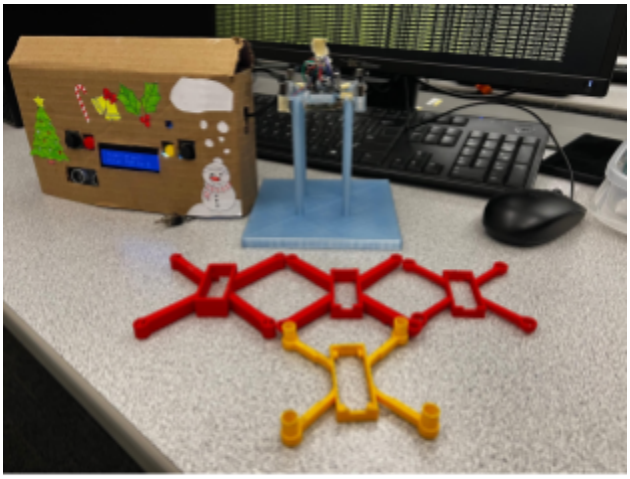
Dronee: Project Writeup

ENSC 351 Froshees

The modes of the drone can be cycled through with the use of the yellow button. Cycling through default mode, ultrasonic mode, and autobalance mode. With the ultrasonic mode, it utilizes the ultrasonic sensor in place of the left joystick to set the PWM speed of all motors. As the user's hand moves closer to the sensor the PWM increases. As the user pulls away, the PWM of the motors decreases. In autobalance mode, the drone solely relies on itself to balance within a desired PWM range set by the left joystick. The self-balancing is done through the use of the Arduino's internal IMU, which is initially calibrated upon power being applied to the drone. When calibrated, it provides accurate pitch, roll, and yaw readings which are then used by the PID to calculate the required PWM motor corrections to level out the drone. These modes, as well as the associated speed manipulations, are displayed on the controller's LCD display for the user to reference.

To shut off the controller program and power down the drone's motors, the red button is used. When pressed, the program sends a packet to the drone signaling it to progressively decrease the PWM value of its motors and disconnect from the controller. During this process, it notifies the user on the LCD, cleans up all the threads used, frees all memory, and ends the watchdog. This is done to prevent an unexpected reboot after the controller program has ended. To fully shut off the controller, the battery switch can then be flipped off to no longer supply battery power to the BeagleBone.

What has not worked well in our system, is the auto balancing feature of the drone. Although it attempts to provide corrections to the drone's motors, it struggles to stabilize itself completely. To try to resolve this issue, we tried to calibrate the PID constants for the auto balancing based on different approaches for determining their values. Although, after many attempts to find the perfect PID constants, we were unable to find the perfect combination to have a perfectly balanced drone. Additionally, we originally envisioned an untethered drone. Due to weight and motor limitations found throughout testing, we realized we were unable to attach a large enough battery while retaining the ability to fly. We tried to solve this through reducing weight in multiple iterations of the drone's frame. Going from the original weight of 20g down to 5g with its final iteration. Even so, we were still unable to carry a sizable battery to have the drone untethered and capable of flight.



Multiple Iterations Of the Drone Design

Dronee: Project Writeup

ENSC 351 Froshees

Important Note: All the code on the target machine (the Beaglebone) and the driver code for the components used with the target is written by the team. For instance, the LCD display's I2C UART BLE drives were written without high level libraries.

Feature Table					
Description	**- Target (BeagleBone) - Other (Arduino)**	**Completeness**	**Code**	**Author(s):**	**Notes:**
Watchdog	T	5	C	Sukha	Adapted from demo_watchdog.c given by professor
Systemd	T	5	Linux Service	Sukha	N/A
Ultrasonic Sensor	T	5	C	Andrew	The sensor often gave unreliable readings.
LCD Display	T	5	C	Andrew	Adapted functions from <wiringPiI2C.h> library
Joysticks/ LED	T	5	C	Sahaj	N/A
BLE Module	T/O	5	C	Sahaj	***Reference Notes Section Arduino end of BLE**
Push Buttons	T	5	C	Bryce	N/A
IMU/PID	O	3.5	Arduino	Bryce	***Reference Notes Section for Further Details**
Motor Driver	O	5	Arduino	Sahaj/Bryce	N/A

Notes/Additional Details:

PID Calculations:

The PID is not properly tuned for balanced flight but does auto-balancing corrections. Additionally we referenced the following code for the PID on the arduino:

- *Ben Finio:* DIY Mini Drone: Arduino™ Altitude Control | STEM Activity

IMU Pitch Roll:

For the IMU on the Arduino we used a library to interface with the LSM6DS3 unit since there was no other efficient way to make use of the onboard IMU unit.

- *Owennewo:* Arduino_LSM6DS3/RollPitchYaw.ino at master · owennewo/Arduino_LSM6DS3
- *Joop Brokking:* MPU-6050 6dof IMU tutorial for auto-leveling quadcopters with Arduino code

BLE for Arduino Nano:

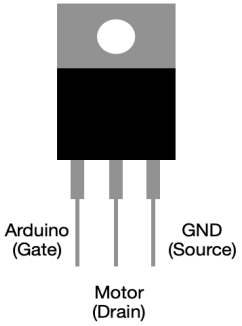
ArduinoBLE example BLE sketch/library was used and modified to suit our needs for the drone since there was no other efficient way to use the onboard u-blox NINA-W102 chipset that contains the BLE component for the board

- *ArduinoBLE Library:* ArduinoBLE library for Arduino

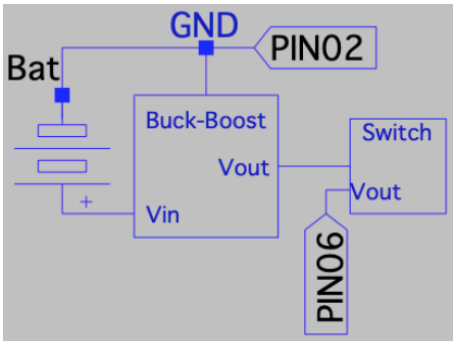
Dronee: Project Writeup

ENSC 351 Froshees

Extra Hardware & Software Used



- **N-Channel MOSFET:**
  4 MOSFETs are used to control the PWM of the motors. The PWM signal from the Arduino allows the motor's speed to be controlled since the Gate Terminal acts as a switch with the ability to adjust voltages between the motor and the battery.
- **3.7V LiPo Batteries (300mAh and 2200mAh):**
  Provides power to the drone and the BeagleBone (controller). BeagleBone operates with 5V, so a buck boost is applied to raise the 3.7V battery to 5V.
- **Buck Boost Converter:**
  The buck boost is used to raise the voltage to 5V to power the BeagleBone since we used a 3.7V LiPo Battery.



- **3-PIN Toggle Switch:**
  Since the BeagleBone is used as a controller, a physical switch is implemented into the design. So that the user can power on/off the controller on demand.
- **3D Printed Frame and Stand:**
  The 3D printed frame is used as a base for the drone. The motors, Arduino Nano and MOSFETs are placed onto the frame. The 3D printed stand is used to calibrate the drone before use.
- **Grove UART Bluetooth Low Energy (BLE) Module:**
  The BLE module is used to send encoded packets which contain information for speed, pitch, roll, yaw, and control mode for the drone.
- **LCD Display:**
  An LCD display is implemented to display important details to the user. Users can check the current control mode, PWM value, or PWM rate of change.
- **I2C Adapter for LCD Display:**
  We used an I2C interfacing board module to utilize and operate the LCD display as the LCD display usually interfaces with the BeagleBone using 16 GPIO pins. We opted to use the module for I2C functionality and efficiency.
- **HC-SR04 Ultrasonic Sensor:**
  The ultrasonic sensor is used to control the drone in the ultrasonic control mode. Users can put their hand on top of the sensor and control the altitude of the drone.
- **Arduino Nano 33 IoT, IMU, and PID:**
  The Arduino was used to receive encoded packets from the controller (using the onboard BLE module) and control the motors of the drone with PWM signals. The onboard IMU was used to obtain pitch, roll, and yaw readings for the drone. These readings were used for our auto balancing feature which utilized a PID providing PWM corrections based on the IMU readings.