Buddy Control Documentation

Paul Kline

January 16, 2015

Contents

1 Supervising

Sending a 'w' ('w' for 'warning') will result in 2 BEEPS. sending an 's' ('s' for 'shock') will result in the attempted pressing of the button. There is a light sensor attached to the shock collar controller that will verify that the light turns on and more importantly turns back off. This is notified to the supervisor by printing "LIGHT ON" and "LIGHT OFF" respectively. Note that while my light detection code is good, it is not guarenteed that the print outs/detection will happen. If the subject appears to have been shocked, they were. Also, a shock command will only activate the actuator at most once every 5 seconds. This safety feature handles the event that multiple supervisors correct the same behaviour at the same time. Therefore multiple shock commands sent in close succession will only shock once.

2 Testing

't' command can be sent to make sure button is being pressed properly when 's' is sent. BUT a 't' command will only perform a shock within the first couple minutes the board is turned on (or reset or program reloaded). This is to ensure that an accidental 't' will not do anything 'bad' when supervising for real. Pressing 't', there is no limit to how many 'shocks' can be performed in a certain amount of time unlike how the 's' command will only shock at most once every 5 seconds as explained earlier.

3 Safety

As stated, the shock command will only be effectful at most once every 5 seconds to prevent corrections from multiple supervisors for the same observed behaviour results in only 1 correction.

There is a light sensor attached to the shock collar controller by electrical tape. The code for this light sensor detects abrupt changes in the lighting environment. Therefore the light sensor code can confirm that the shock command was successful, but more importantly activates a fail-safe... IF the light sensor detects that the button has stayed on for longer than intented (a very short amount of time–slightly longer than it takes to press the button), this indicates that somehow the button has become stuck down. We then call panic() which will attempt to remedy the situation. Right now panic() just pull the servo back to its resting position and detaches. Once the other parts come in, panic() will also somehow attempt to free the controller. The mothod of freedom may be permanent or not, I haven’t decided how severe to fail (since it is possible to trigger a fail-safe without actually pushing the button–i.e. a new bright light is introduced suddenly. But this should not happen since the light sensor is mostly covered by electrical tape.

We panic() at most 10 times (each time checking to see if the condition has been corrected or not of course). If after 10 times, we stop panicking. It clearly isn’t working, or the button isn’t actually depressed. Then after 15 seconds, if we still detect the problem, we assume that this is just a new (and quickly introduced) lighting environment and re-calibrate this new lighting level as indicating the ’off’ light state.

4 How to detect a button press

We keep track of the DIMMEST light that has been detected within a certain amount of time (I believe currently 700 milliseconds). If at any time, the dimmest value recorded within 700 milliseconds is less than 300 less (on a scale of 0 - 1023), this is considered a drastic light change, characteristic of a button press. Once we detect that the button has been pressed, we make note of the time that occurred. If the light level does not get dimmer by at least the same threshold that triggered the button on state within (currently) 700 milliseconds, we trigger the panic. Let it also be noted that during the button-on state we continuously look for and keep the brightest value recorded. It is then that brightest recorded value we wait for the jump back down from.

In this way, we can handle slower light changes with ease (i.e. sunlight) for we only trigger the change in light level recorded.