Light Detector Monitor

Description:

This project involves building a monitoring system for the light detector that my group is building. By reading the signals that are going into and being outputted by the light detector, this monitor is capable of detecting if there is a fault within the light detector and notifies the user.

Summary of Design Challenges/Why This Is An Interesting Problem:

In creating our group project, much of the work included building the circuitry itself, the software to make the components work together, and the tests for the software itself. Beyond testing the software itself, there does not appear to be a definitive way of testing the hardware itself. By creating this monitor, the signals going into and out of the system can be read and interpreted. Depending on the signals, certain behaviors can be expected out of the system and if these behaviors do not occur, then we know that there are faults, either in the software or hardware. Building a monitor for the system is an additional method of testing the integrity of the system.

The design challenges in building this monitor centered around being able to read the system’s signals within a reasonable amount of time while also deciding what to do with these signals to make the monitor as comprehensive as possible. The monitor cannot expect for changes to happen instantaneously so the monitor needs to be able to monitor incoming signals and any expected future signals without blocking any check functions. The monitor also needs to be able to take into consideration the various situations that the system could find itself in, such as if the lights do not change or if the lights are changed manually by the user.

Approaches Taken/Challenges Encountered:

On the software side, the most prevalent challenge has been figuring out how to make each check not be too sensitive. If the monitor anticipates for the room’s lights to change, the change in lighting is not instantaneous. The monitor also has its own photoresistor for redundancy; the photoresistor on the monitor might have a mismatch compared to the system’s photoresistor for a split second and falsely indicate a fault in the system’s photoresistor. The initial approach taken has been to have each test run with a delay function: each check began when certain conditions were met, delay, and then real world results were compared with expected results. Eventually, too many delay functions made the monitor inaccurate since an appropriate monitor check might not be triggered during a delay. This issue was fixed by running a timer without blocking the other functions. My approach taken for this fix involved running each check within the loop function, setting a flag when a check should be run, and beginning a timer using the millis function when the monitor receives a certain input. If a sufficient amount of time has passed, the check is run - if the check fails, a fault is indicated, and if the check passes, the flag is reset and the monitor continues running.

For the signals coming out of the system, they should ideally be handled as soon. For example, the system had a button that would invoke an interrupt service routine (ISR). The other main signal of concern was when the servo motor would be activated since that signal was brief. In order to capture these signals quickly and accurately, these system pins were connected to the Arduino as interrupts. These interrupts would set off a non-interrupt timer and a flag, where the Arduino would perform the appropriate check when a sufficient amount of time has elapsed, after which the flag gets reset.

A general challenge in designing a monitor has been determining which checks should be run. At a basic level, the individual components can be checked, such as having the system's photoresistor be compared against the monitor's photoresistor to ensure that the photoresistor is running correctly. The more challenging aspect has been creating the more comprehensive checks, such as expecting the motor to go off when claps are detected or expecting for the lights to change when the motor goes off. In figuring out which comprehensive checks should be run, I had looked over the system's code and FSM, trying to anticipate which conditions switched or did not switch the current state of the system.

Diagram:

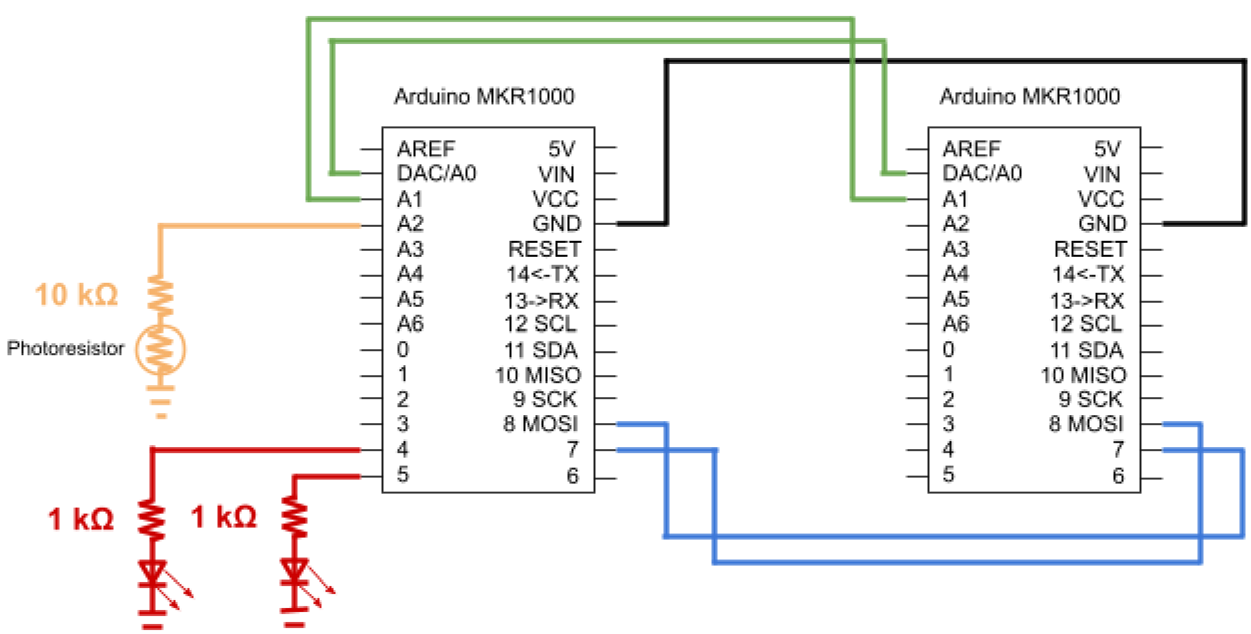

Photo: