Intelligent Indoor Growing Module

Nathan Worthen, Canyuan Tian, Jacqueline Lundy

Chaz Granholm, and Gustavo Santi Pereira Neis

Illinois Institute of Technology

# Abstract

Project goal was to devise a monitoring and control system for indoor food production.  Consideration of space, cost, and ease-of-use were taken while providing the most efficient method to grow fresh and healthy produce.  The prototype hydroponic system constructed of common PVC tubing at low angles delivers a gentle flow via gravity.  Analog photocells provided sensor data to the microcontroller system to display to the user and toggle artificial grow lights. Due to the high amperage spikes inherent to compact florescent bulbs, the group had to alter their methods and utilize the Arduino Platform rather than the Intel Galileo originally intended.  As the Arduino does not natively support networking, the plans for working with the Plotly beta were pushed back.  Relevant data is output to user in real time via scrolling liquid crystal display.

The end result augments the efficient hydroponic growing technique with the most cost efficient utilization of natural energy when possible.  The incorporation of further sensors and analytic software is possible, as are various forms of power.  Thus, the flexible nature of the custom embedded system will allow for a multitude of additional hardware, software, and infrastructure alternatives in the future.

# Contents

[Abstract 2](#_Toc386621929)

[Introduction 4](#_Toc386621930)

[Structure 4](#_Toc386621931)

[Light Sensor 5](#_Toc386621932)

[LCD 5](#_Toc386621933)

[Observations 6](#_Toc386621934)

[Relay Amperage 6](#_Toc386621935)

[SD/OS 7](#_Toc386621936)

[NEXT STEPS/ IMPROVEMENTS 8](#_Toc386621937)

[References 9](#_Toc386621938)

[SimpleTimer Library for Arduino 9](#_Toc386621939)

[LiquidCrystal Library for Arduino 11](#_Toc386621940)

[Beefcake Relay Control Kit 13](#_Toc386621941)

[Mini Photocell 13](#_Toc386621942)

# Introduction

## Structure

The design implemented in this prototype is actually of little significance.  To explain, this planting medium in its very nature is intended to be scalable to the demands of its environment.  For homeowners with the intention of placing the medium outside, something that allows the plants to be exposed to more sunlight may come in the form of a more sprawling horizontal shape.  Likewise, an urban farmer may decide between placing the medium in the sill of a bay window, or entirely away from natural light while focusing on artificial grow lights.  For purposes of this prototype, we chose the former of the two.

The advantage of utilizing the design as we have implemented becomes clear for urban dwellers wishing to produce their own edibles.  The profile is slim enough to fit upon a window sill, yet maintains nearly 30” of horizontal sprawling distance in order to collect sunlight.  The second row, as seen below, is utilized to maximize growing space while maintain at least partial sunlight exposure.

        The infrastructure was built utilizing 2” Polyvinyl chloride (PVC) pipe.  As seen above in Figure 1, a series of 90° elbow fittings twisted at such an angle to provide plants a maximum of 17” height before reaching the next layer of pipe.  Next, two holes were drilled on each section of 24” pipe separated by 8” of thinning distance.  Plastic baskets containing rock wool are then placed inside each hole with accompanying seeds in order to produce the crop.

        Lastly, the entire structure rests upon a water basin measuring 21” x 31”.  The lid of the container has been drilled in order to allow a final PVC elbow to point downwards, thus allowing running water to continually flow back into the basin.  Water is continuously pumped from the basin to the top of the structure utilizing a submersible 12W water pump.  The pump chosen for this scenario is adjustable as needed in order to supply a constant stream of water through the planting medium without being at such a pressure to cause harm to the plants.  The water in the basin is laced with an organic, liquid fertilizer in order to supply the needed nutrients to the plants.  While it is necessary for the water pump to run daily without pause, its power consumption is minimal.  At the relatively high price of KWh in urban Chicago, annual power consumption of the pump amounts to just under $14 per year.  This cost, if desired, could be minimized by utilizing alternative power sources, such as solar.

## Light Sensor

The light sensor was implemented to save energy of the prototype in its period of function. This period is given by the program implemented in Arduino to provide an equivalent of 17 hours sunlight. Throughout the next 7 hours, the lamp and the sensor will not function.

In the prototype a photoresistor(LDR) was used to analyze the amount of light emitted by a source, and according to the program implemented shows the percent of light readed by the photoresistor. This reading happens every 10 minutes following the program’s clock, therefore it’s analyzed if the amount of light received surpass 50% or not (following the patterns selected in the program). If the reception of light is inferior to 50%, the lamp that lightens the plants will keep functioning. If is superior to 50%, the lamp will turn off and plants will use natural lighting for photosynthesis.

The photoresistor will be located near to a natural source of light, in this case the place chosen was a window. The sensor is linked to the main system, and is controlled via Arduino.

## 

## LCD

  The function of the liquid crystal display (LCD) is to print out information that may be helpful to the product’s user.  One example may be the current natural light value as detected by the analog sensor and the duration the light has been on or off.  The data read in by the analog sensor is passed to the LCD on a continual loop, allowing the user to maintain awareness of the values.  The range of output may vary between 0 and 1000.  For our prototype, when a value of natural light falls lower than a threshold of 550, our light will be turned on automatically.  After the light turns on, a timer is displayed on the LCD with the duration the light has been on.  The same logical process takes place when the light shuts off.  This data is intended to keep the user informed of light readings, as well as allow them the opportunity to adjust the planting medium’s position in order to maximize natural light absorption.  Due to the 20x4 pixel size of the display screen, all the information desired to be shown is not visible at once.  Therefore, our microcontroller implements the scrollDispayLeft() method in order to scroll the data across the screen for the user.

# Observations

## Relay Amperage

One struggle that the team faced when building the prototype was the low amperage given by Galileo to the system. The lamp needed a beefcake relay control (model KIT - 11042) to be connected to the jack. The minimal amperage to make the relay control works is 5A, and the system was only giving 3.7A. The plugin that feeds Galileo (phihong switching power supply, model PSC12R-050) wasn’t letting the system receive all the 5A that the relay control needs, and in the first moment we couldn’t make it work correctly.

It was also noticed that the Arduino when used by USB worked fine. Otherwise the same plugin that was used on Galileo didn’t work on Arduino as well. Even getting success using Arduino, one of the objectives aimed by the prototype was the use of Galileo, so a research was made to solve that problem to keep working with Galileo system.

By analysing the structure of the microprocessor Galileo, it was possible to solve the problem of low amperage to active the relay control. That was due to the jumper (VIN jumper) on Galileo, that when removed, it allowed an amperage superior to 5A. Even after removal of the jumper, the problem persisted, and because our lack of time the team had to use Arduino as its final choice.

## SD/OS

Another struggle our team faced when using Intel’s Galileo was keeping a persistent sketch on the board after unplugging the USB serial port as well as the outlet power supply.  The only way this board was able to retain a sketch was through the use of a micro SD Card.  Even with the card, if the outlet power supply were detached, the sketch would no longer be available.

The Intel Galileo is a Linux-based Arduino compatible development board that’s driven by Intel’s x86 architecture.  Although it has the ability to run the Arduino IDE, in order to perform software or file-system maintenance on the board, you must utilize the Linux command-line terminal or upload additional software.  It is published that Intel architecture features between 256KB and 512 KB dedicated for sketch storage, but the installation of firmware updates maybe the reason this storage was unavailable for use for our sketches.   In order to save sketches to the SD card after powering down, the Galileo must boot from the SD card with the LINUX IMAGE FOR SD for Intel Galileo installed on it.

# NEXT STEPS/ IMPROVEMENTS

With a total of 20 Analog and Digital I/O Pin availability, the Arduino Uno enables this system to be highly scalable.  Adding additional sensors to measure temperature and ph balance would be a logical next step.  Potential power source alternatives, like solar power can be explored as well in order to store energy for our system in case of a power outages.  Also, through the use of an Ethernet Shield Expansion Board, networking and wireless capabilities can be considered.  This feature could allow for real-time online data monitoring from our sensors to the World Wide Web.

There are a number of services on the Internet that offer Live Streaming Data Monitoring capabilities.  One such service that our team came across is called Plotly, a free online analytic and data visualization tool providing multiple API’s and data graphing capabilities.  With the use of services like these, data can be accessed from any device connected to the Internet.

# References

## SimpleTimer Library for Arduino

Author:  Marcello Romani

Contact:  mromani@ottoecnica.com

License:  GNU LGPL 2.1+

**Getting Started:**

Download Library From: [www.playground.arduino.cc/Code/SimpleTimer](http://www.playground.arduino.cc/Code/SimpleTimer)

**Installation:**

1.       Create a folder named SimpleTimer in your sketchbook/libraries folder.

2.       Copy and paste the code for SimpleTimer.h and SimpleTimer.ccp into your favorite text editor, and save those two files under the SimpleTimer folder you just created.

3.       Launch or restart the Arduino IDE to let it detect the new library.

4.       Copy and paste the example code to get you started with the library.

**Usage:**

            #include <SimpleTimer.h>

            SimpleTimer timer;

            void setup(){

                            timer.setInterval(1000, f);

            }

            void loop() {

                            timer.run();

            }

            void f() {

                            // do something

            }

            timerID = timer.setInterval(1000, f);

            With this constructor, function f is called every 1000 milliseconds.

## LiquidCrystal Library for Arduino

Author:  Public Domain

Contact:

License:  GNU LGPL 2.1+

**Getting Started:**

Download Library From: [www.playground.arduino.cc/en/Reference/LiquidCrystal](http://www.playground.arduino.cc/en/Reference/LiquidCrystal)

Note: This library is included in Arduino 10.5

**Installation:**

1.       Include the file header at the beginning of the Arduino sketch.

2.       Initialize a LiquidCrystal object.

**Usage:**

            #include <LiquidCrystal.h>

            LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

 // set up the LCD's number of columns and rows:

                            lcd.begin(16, 2);

                            // Print a message to the LCD.

                            lcd.print("hello, world!");

}

void loop() {

                            // set the cursor to column 0, line 1

                            // (note: line 1 is the second row, since counting begins with 0):

                            lcd.setCursor(0, 1);

                            // print the number of seconds since reset:

                            lcd.print(millis()/1000);

}

            This example is included in the LiquidCrystal Reference Library.

## Beefcake Relay Control Kit

Producer:  Shaanxi Qunli Electric Co., Ltd.

Contact:  [sxqlljd@hotmail.com](mailto:sxqlljd@hotmail.com)

**Getting Started:**

Product page available at: www.sparkfun.com/products/11042

**Requirements:**

1.       Microcontroller capable of outputting a 5V power source, a 5V pulse digital signal, and a grounded connection.

2.       A device to output the switched power to, capable of receiving a maximum load between 30A at 240V or 8A at 120V.

3.       Basic Arduino coding knowledge.

**Usage:**

Due to complexity and safety concerns, please refer to the Sparkfun tutorial located at: [www.sparkfun.com/tutorials/119](http://www.sparkfun.com/tutorials/119) for further usage information.

## Mini Photocell

Distributor:  sparkfun.com

Contact:  [customerservice@sparkfun.com](mailto:customerservice@sparkfun.com)

**Getting Started:**

Product page available at: www.sparkfun.com/products/9088

**Requirements:**

1.       Microcontroller capable of reading in an analog signal.

2.       A 10k Ohm resistor.

3.       Basic Arduino coding knowledge.

**Usage:**

int LDR\_Pin = A0; //analog pin 0

void setup(){

            Serial.begin(9600);

}

void loop(){

            int LDRReading = analogRead(LDR\_Pin);

            Serial.println(LDRReading);

            delay(250);

}

In the previous example, a miniature photocell is utilized to read in an analog signal and print the value to the console.  The photocell is in fact a light dependent resistor (LDR) which varies its resistance based upon the light saturation it receives.  This fluctuation of resistance allows the microcontroller to quantity a relative value to read in serially based upon the resistance.  It is noteworthy, however, that this LDR is not very precise and is unreliable when calculations of LUX or lumens are required.