Smart-Toaster

ELEX 4330 Technical Project

10/24/2011
BCIT – Electrical and Computer Engineering Computer/Control
Jin Won Seo, William McLachlan, Raymond Chan

OBJECTIVE AND EXPECTED BENEFITS OF THE PROJECT

The Primary objective

The primary objective of the project is to design and implement a toast, Smart-Toaster® which has remote control and sensor functions. In addition, Smart-Toaster® allows the users to be able to check a status of toast of bread so that the users notice how long the time is remained to be done and the overall process of toasting.

The Second objective

Through the project, we can apply and solidify knowledge that we have been learning as an electrical student such as analyzing and implementing circuitry, handling a micro-controller and expanding basic program skills to web-based program that has not been taught. Moreover, we are capable of handling each interface between each component such as between the toaster and micro-controller, and using micro-controller as a web server, and webpage to the user while doing the project. Even though dealing with the sensors and web based programming are new to us, using the knowledge that we have been learned will help us to approach new technologies.

The Benefits of the project

1. Adding a remote control function using TCP/IP enables the users to toast anywhere and anytime.
2. Improved sensors make the toaster smarter so that unlike regular toaster it can more fit to the users' taste.
3. GUI provides detailed status of how well toast is done and how long takes and remained to be done, which make the users easy to handle the toaster.
4. Unlike other toasters, the users do not need to push the plunger to load and start toasting. Automatic loading and unloading mechanical plunger releases this trivial effort.

FEATURES

Remote controls

Only remote control functions are allowed to users in the scope of project; in other words, the users are not allowed to set the start time, degree of toasting, and check the status.

1. In order for the users to control and handle Smart-Toaster®, the users can only access through the GUI.
2. The GUI has two functions.

- ***SETTING [FIGURE 1, GUI DESIGN,P4]***
    - This menu provides the users to choose start time.
    - *The users are able to set up the start TIME (year, month, days, hours, minutes, and seconds) using select buttons provided by GUI*.
    - In addition, this menu allows the users to choose the degree of toasting or desired doneness using select button (e.g. light, medium, dark).

- ***STATUS [FIGURE 2, GUI DESIGN, P4]***
    - When the users choose the status menu, the users can easily notice the remained time to be done toast.
    - Also the status bar provided by GUI tells the users how much toasting is done graphically.

Mechanical operations

1. Solenoid attached to toaster's plunger pulls down and starts the toasting process when controlled by the micro-controller.
2. On-Off switch turns the Smart-Toaster® on and off.
3. RJ-45 connector for network connection to a webpage to deliver the settings and status information to the user.
4. Enclosure will house the microcontroller, attached to the toaster but isolated from the heat.
5. LEDs indicate if Smart-Toaster® is on or off or if it is in the process of toasting.
6. *Smart-Toaster® loads bread into a slot at the start time and unloads at the end time automatically. (if time permits.)*

Sensors

Smart-Toaster® has two sensors to control how toasted bread is based on the users' choice as it is set through GUI.

1. Colour sensor detects how toasted bread is.
2. This sensor's colour data is read by the micro-controller to determine doneness.
3. Thermometer detects the temperature of the Smart-Toaster unit to prevent overheating caused by extended operation that might lead to damage or safety hazards.

Interfaces

As it shown in [Figure 2, system block diagram], the interfaces are divided into three main parts.

1. BETWEEN TOASTER AND MICRO-CONTROLLER

- Toaster unit is connected to the *Arduino* microcontroller to be controlled and operated by micro-controller.
- Micro-controller attempts to retrieve status data by sensors such as the temperature and colour of bread and according to the data, the controller will toast the bread to the perfect desired level.

2. BETWEEN MICRO-CONTROLLER AND NETWORK

- Once the users set the start time and how dark they want their toast on the webpage , the micro-controller reads it over the network and uses it to determine how it operates.
- To be able to provide the status information, the micro-controller reads the colour sensor to gauge the status of the toasting job.
- Arduino micro-controller module set up to act as the server to deliver and retrieve all data over the network needed.

3. BETWEEN WEBPAGE AND USERS

- The user attempts to access the server to either set up start time, how dark they want their toast or check the process of toasting via GUI.
- Once the users set the above information, the server will read the input and operate the toaster accordingly.
- Status updates such as current level of bread doneness and estimated time remaining are reported on the webpage.

METHODOLOGY

SUMMARY

- Obtain Parts
    - *Arduino Uno* microcontroller
    - *Arduino Ethernet Shield* Ethernet Interface
    - *ColorPal* Colour Sensor
    - *Toaster*
    - *Solenoid and/or other actuators*
- Research & Learn to Use Components
    - Programming/debugging Arduino
    - Using Colour Sensor
- Write software for Microcontroller
    - Interface to sensors
    - Communicate via ethernet
    - Set up web server
    - Write toast algorithms
    - Control Mechanisms
- Design and Build Mechanisms
    - Toaster plunger actuator
    - Control heater element (on/off)
    - Bread loader (*optional*)
- Design and Build Enclosure
    - Hold sensors
    - House Arduino
    - Power input

DETAILED BREAKDOWN

ARDUINO W/ ETHERNET SHIELD WILL BE SET UP AS WEB SERVER.

- Research and implement basic I/O to/from arduino to web page.
    - Start with I/O to webpage, basic
    - Move up to Full Features.
- Program the web interface.
    - Input to Toaster
    - Output from Toaster

COLOUR SENSOR WILL SEE HOW TOASTED THE BREAD IS.

- Research and obtain data on typical output values for different degrees of toastedness.
- Interface colour sensor to arduino.
    - Physical Connections
    - Software Implementation
- Write algorithms to use data obtained to determine when to turn heaters off & pop up toast.

WRITE SOFTWARE FOR MICROCONTROLLER

- Gather data from sensors
  - Colour Sensor
  - Temperature Sensor
  - “Toast loaded” Sensor
- Process data
  - Determine how toasted toast is from colour sensor data
- Interface to Web
  - Web server setup
  - Input from web
  - Output to web
- Control Mechanisms
  - Activate/deactivate toaster plunger
  - Control On/Off of heating element
  - Output to status LEDs

HEAT SENSOR AS SAFETY DEVICE

- Make sure toaster is not on for too long.
- Make sure toaster does not get dangerously hot.

TOASTER PLUNGER WILL BE ACTUATED BY SOLENOID/MOTOR.

- Design a mechanism for raising & lowering the plunger.
- Design interface/control to microcontroller.
  - Physical Connections
  - Software Implementation
- Build and test mechanism.

BUILD ENCLOSURE FOR SMART-TOASTER

- Design enclosure to house micro and sensors.
  - Power Input
  - On/Off switch
  - Indicator Lights
  - Network cable jack
  - Micro and sensors stable and secure.
- Build enclosure. Finish Physical Toaster Unit.

LOAD AND UNLOAD BREAD INTO TOASTER

- Controlled by microcontroller
- Loads Bread
  - Picks up bread
  - Drops into toaster slots
  - Tells micro when bread has been loaded
- Unloads Toast
  - Picks up toast out of slots
  - Moves toast above plate
  - Drops toast

SYSTEM DIAGRAM

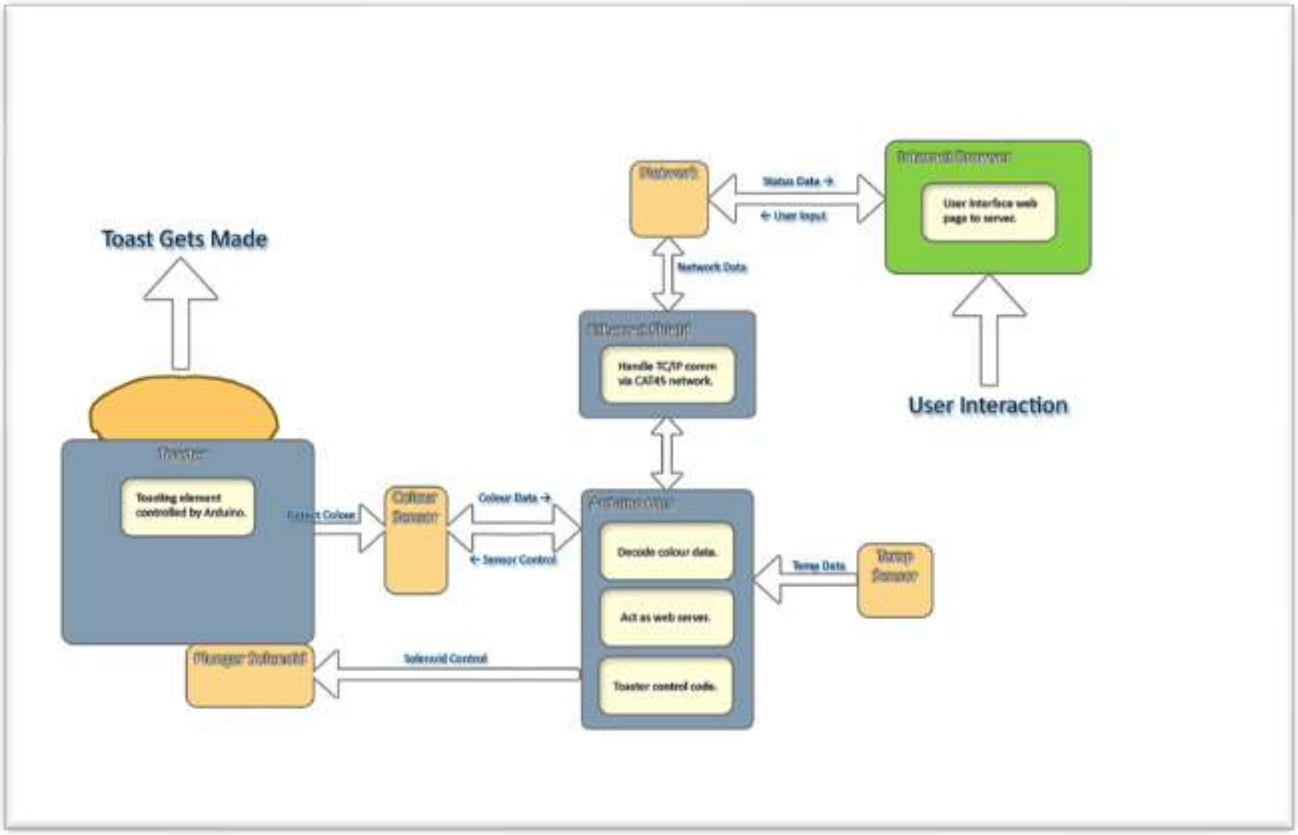
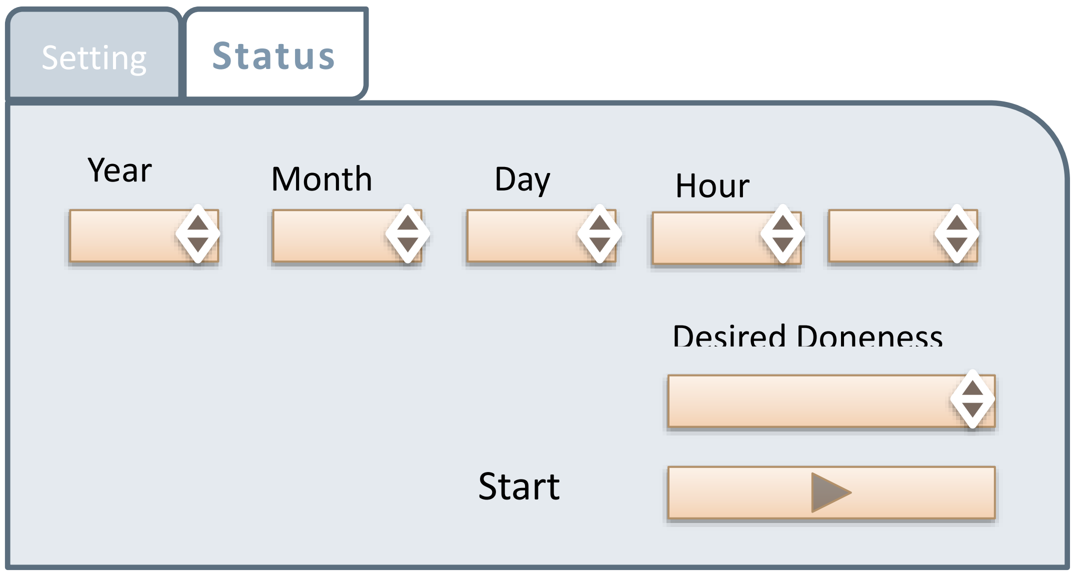


Figure 2- System Block Diagram

1. Arduino within the Smart-Toaster® acts as a server
   - The server updates the webpage with status information.
   - Reads input data from user on webpage.
2. Colour Sensor looks at toast
   - Detects colour of toast
   - Sends colour data to Arduino
   - Arduino decodes colour data into *"toastedness"* to determine when to stop toasting
3. Solenoid Controls Toaster's Plunger
   - Pulls bread down
   - Starts heating element
4. Webpage interfaces the Smart-Toaster to the User
   - see *Webpage Design* for UI layout.
   - Input/Output for user
5. Temp sensor acts as a safety feature
   - Arduino reads temp to decide whether to turn off unit when too hot.

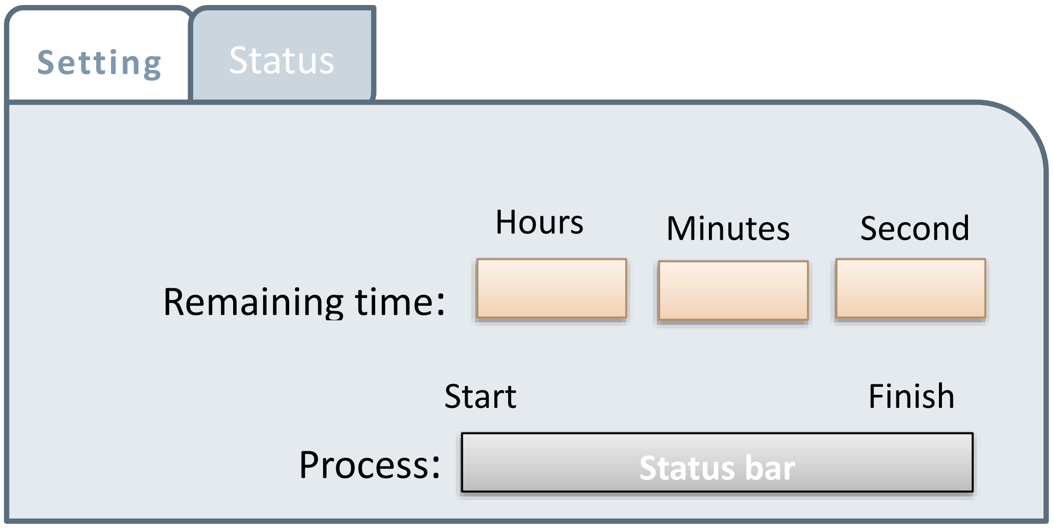
WEBPAGE DESIGN

GUI design (select the setting menu)



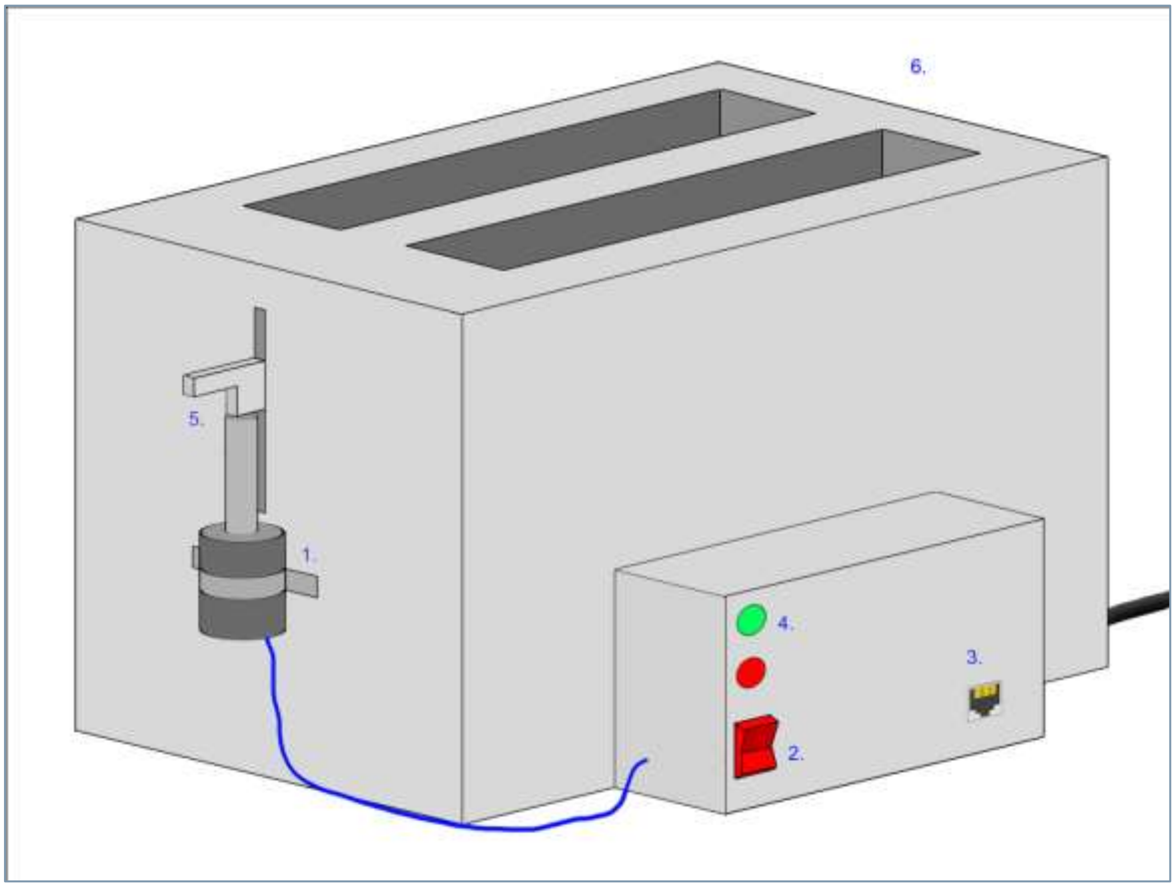
[Figure 1], GUI Design

GUI design (select the status menu)



[Figure 2], GUI Design

PHYSICAL DESIGN



[Figure 3], Physical design

1. Solenoid set up to control toaster's plunger.
2. On-Off switch controls the Smart-Toaster.
3. RJ-45 jack for network connection.
4. On-light and toasting light indicate operation.

   Enclosure mounted onto side of toaster.

5. Solenoid attached to toaster's plunger.
6. Toaster's original timing circuitry removed.

FLOW CHART(WEB PAGE)

Main GUI (setting.jsp or .php)

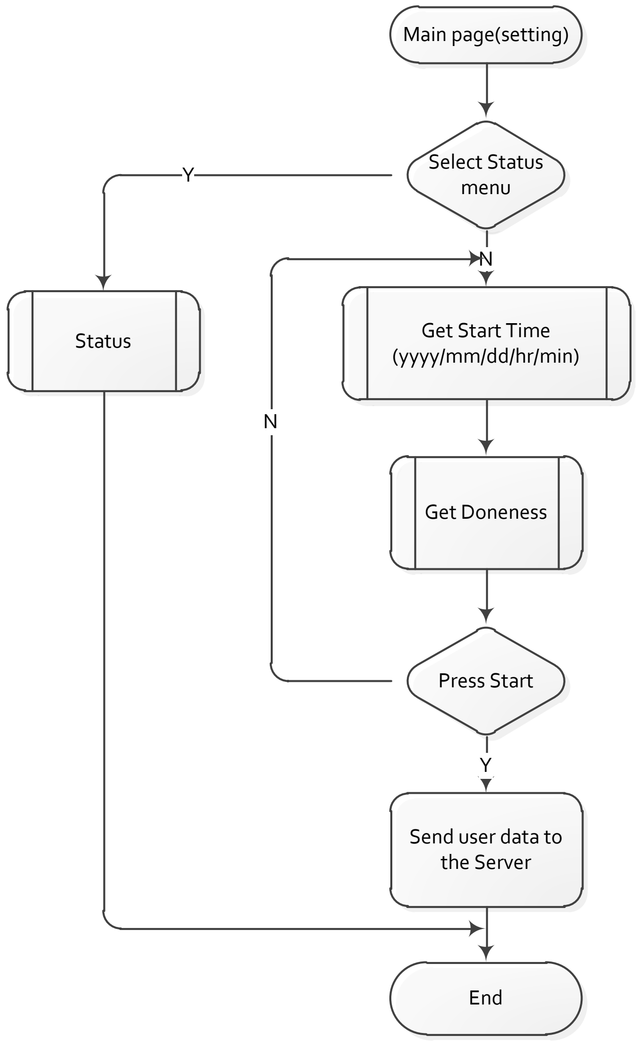


Figure xxxx

1. The users are allowed to access main page which is setting.jsp. If the users want to check the status, accessing to the main then click the status menu. The status flow chart will be followed.
2. Setting.jsp(or .php) requires the users to set the start time(yyyy/mm/dd/hr/min) as well as the desired doneness.
3. Once the users press the start button on the page, the user's data will send to the web server.

HTTP (get)

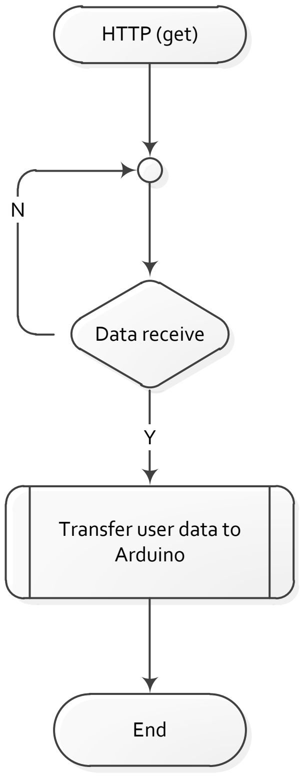


Figure xxx

1. The protocol between the users and the server follows HTTP (TCP/IP).
2. Once the start button is pressed, the server will be sent the data and transfer the data to the micro-controller (Arduino).

Status (.jsp or .php)

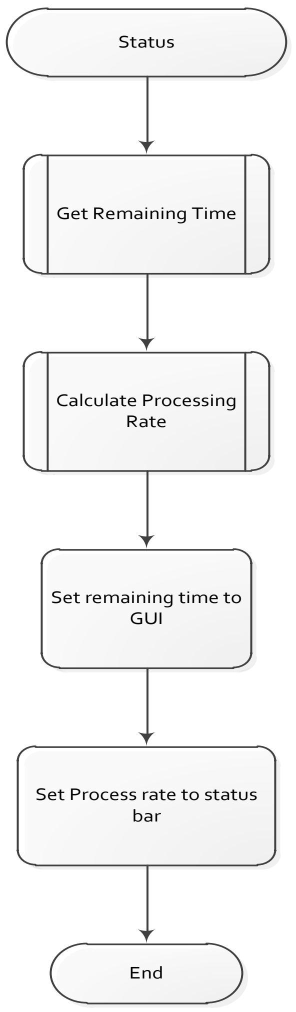
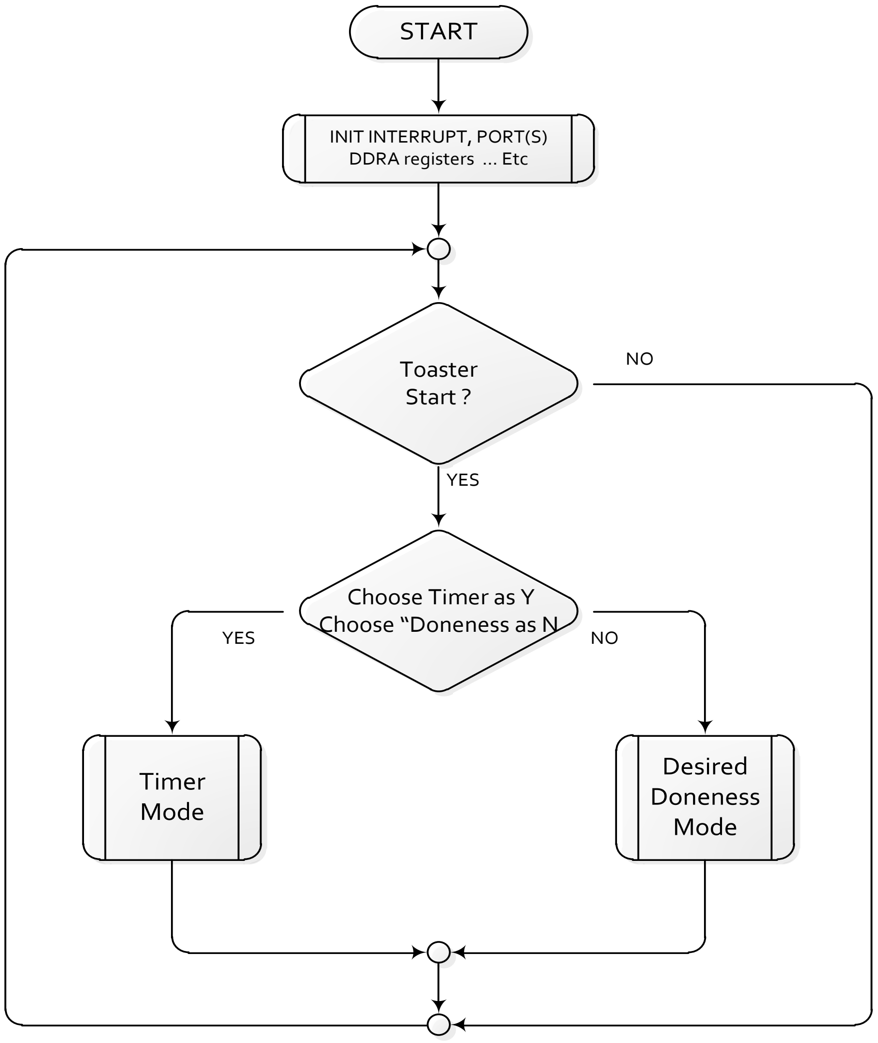

Status
Get Remaining Time
Calculate Processing Rate
Set remaining time to GUI
Set Process rate to status bar
End

Figure xxxx

1. This web page only allows to access via main page (setting.jsp or .php).
2. As the users press the status menu in main page, status.jsp or .php tries to get the status data which is the remaining time and the how far the process is done from the micro-controller.
3. Once the status data are sent from the micro-controller, the modules inside this page attempt to manipulate the data in order for the user to see the proper format (yyyy/mm/dd/hr/min/sec and progressive bar).

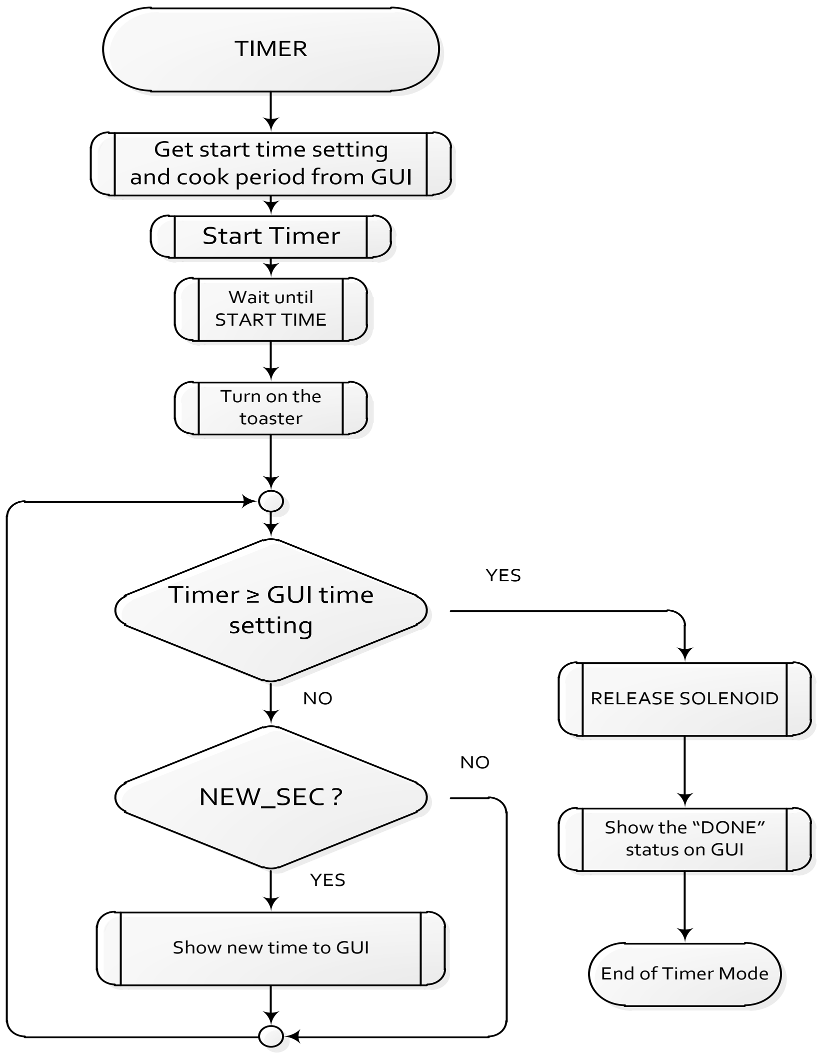
FLOW CHART(ARDUINO)

Main Arduino (selecting timer mode or doneness mode)



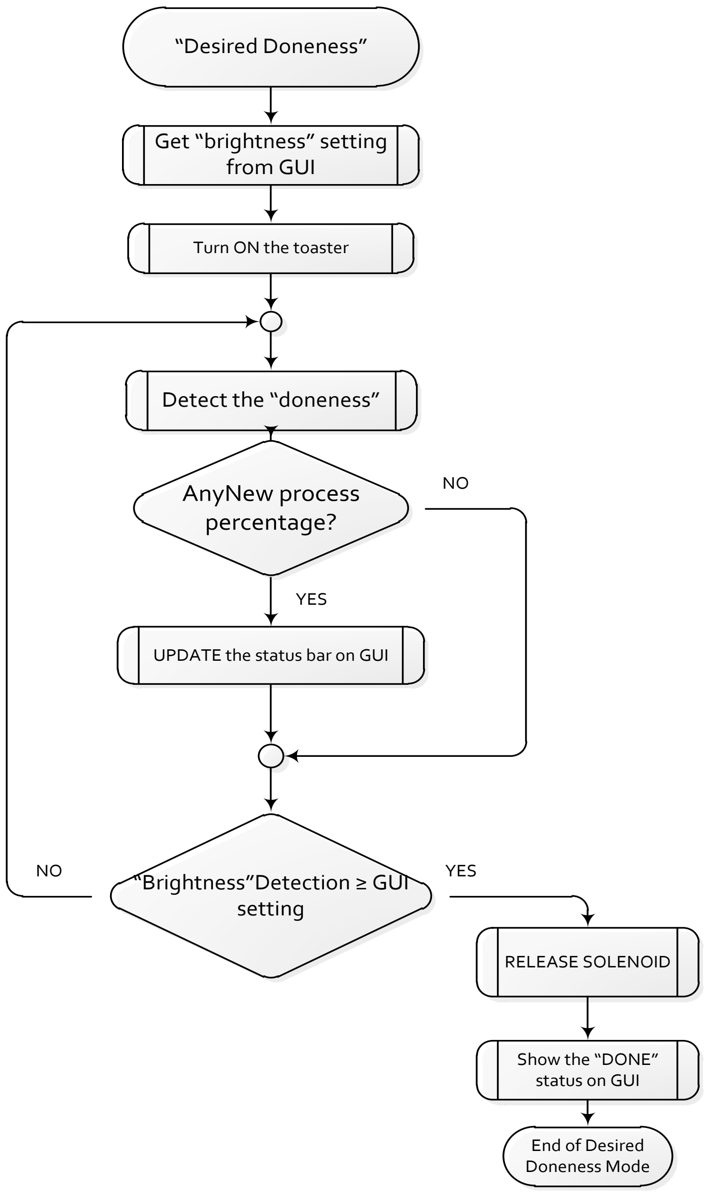
1. The arduino initial all port and register in order to receive the data.
2. The toaster will stay off until it receive "turns on" signal from the GUI.
3. It enters either "TIMER MODE" or "Doneness MODE" to start working.
4. After it has done the job, it waits for next task.

TIMER MODE



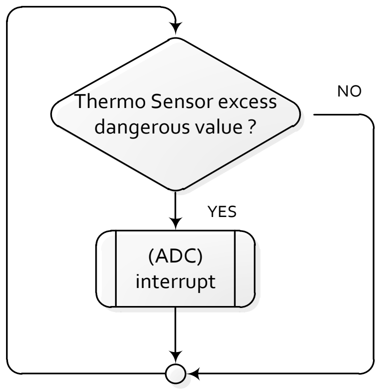
1. The arduino gets the time setting and cook period from GUI.
2. It waits until the start time.
3. It starts to turn on the toaster for the cooking period which it gets from GUI before.
4. It shows the remaining time during the toasting period.
5. After it finishes the task, it turns off and signals to the GUI.

DONENESS MODE


"Desired Doneness"
Get "brightness" setting from GUI
Turn ON the toaster
Detect the "doneness"
AnyNew process percentage?
NO
YES
UPDATE the status bar on GUI
"Brightness"Detection ≥ GUI setting
NO
YES
RELEASE SOLENOID
Show the "DONE" status on GUI
End of Desired Doneness Mode

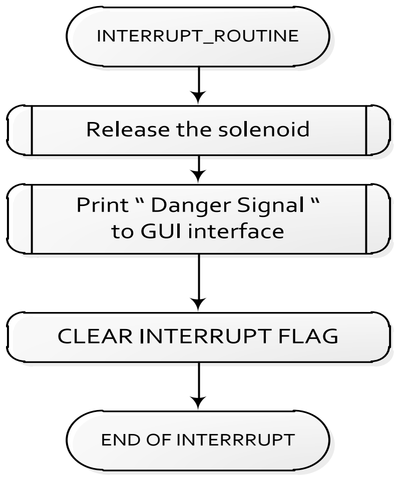
1. It gets "bread darkness setting data" from the GUI setting.
2. It turns on the toaster and detects the darkness of the toaster.
3. The brightness data to compare the "GUI setting data" in order to determine if keeping working or not.
4. When the darkness reaches the setting, it turns off the toaster and signal to GUI.

Interrupt Initialization



1. Detect the temperature data in ADC.
2. if it exceed the dangerous temp, arduino goes to interrupt mode.

InterruptRoutine



1. Release the solenoid immediately.
2. Signal to GUI the toaster exceed the temperature.
3. Clear the interrupt flag

4.

BLOCK DIAGRAM

Web

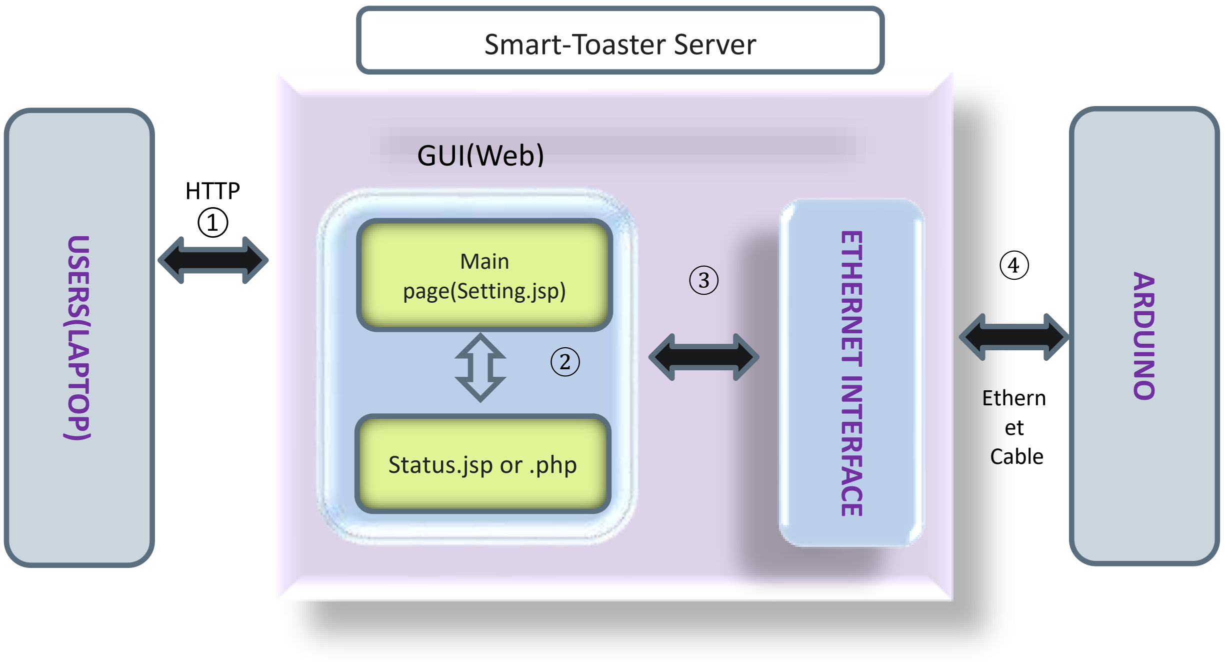


Figure xxxx

① The users try to access to the smart-toaster server, which is setting.jsp or .php.
② Through the main page, the users are to access to the status page.
③ Once the users either set the start time and press the start button or check the status, the web server attempts to access the micro-controller (Arduino) via the Ethernet interface provided by Arduino.
④ The web server and Arduino communicate through the Ethernet cable.

Ardiuno

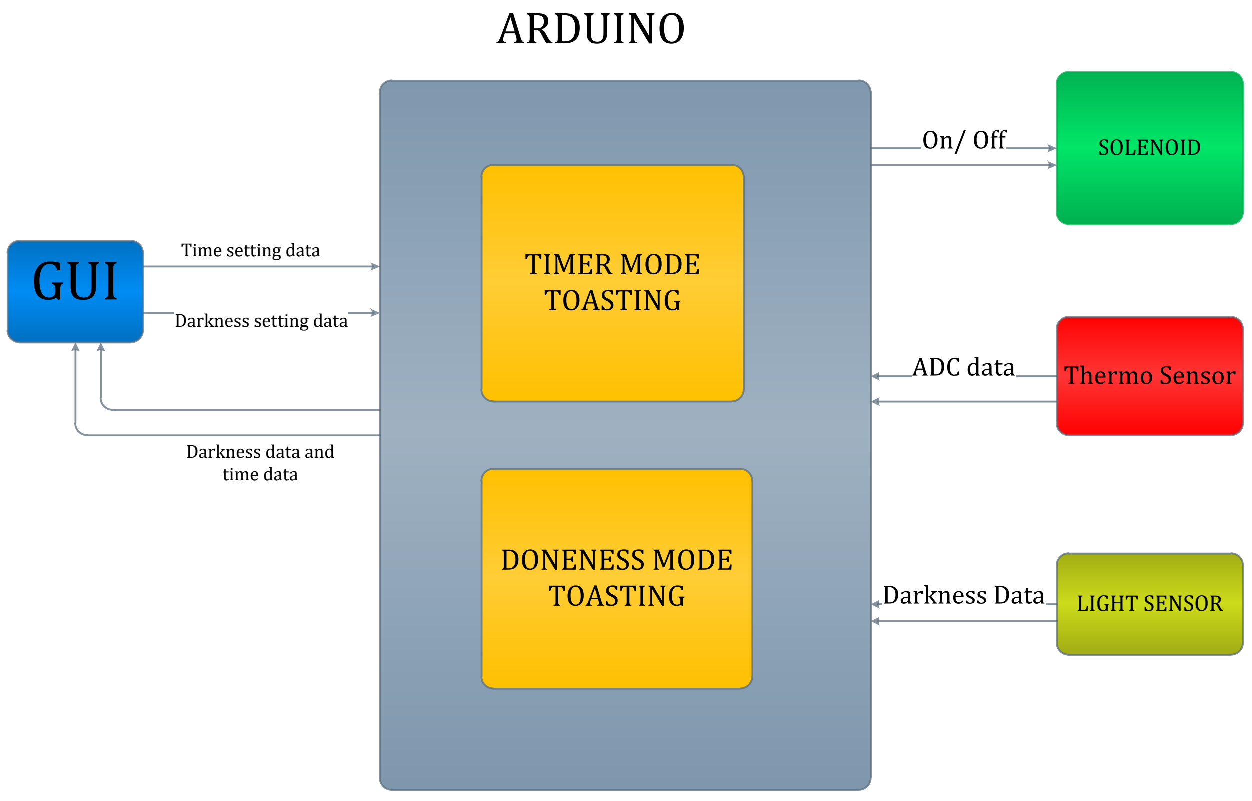


Figure xxxx

1. The arduino uses getting “MODE data” from GUI, to choose the working mode.
2. In the order way, the arduino send the darkness data and time data back to GUI.
3. Thermo sensor send volt to ADC, as the controller detects temperature data, and do interrupt if necessary.
4. The program controls the solenoid to turn on / off
5. Light sensor detect the Darkness of the bread and send it to arudino.

DIVISION OF LABOUR

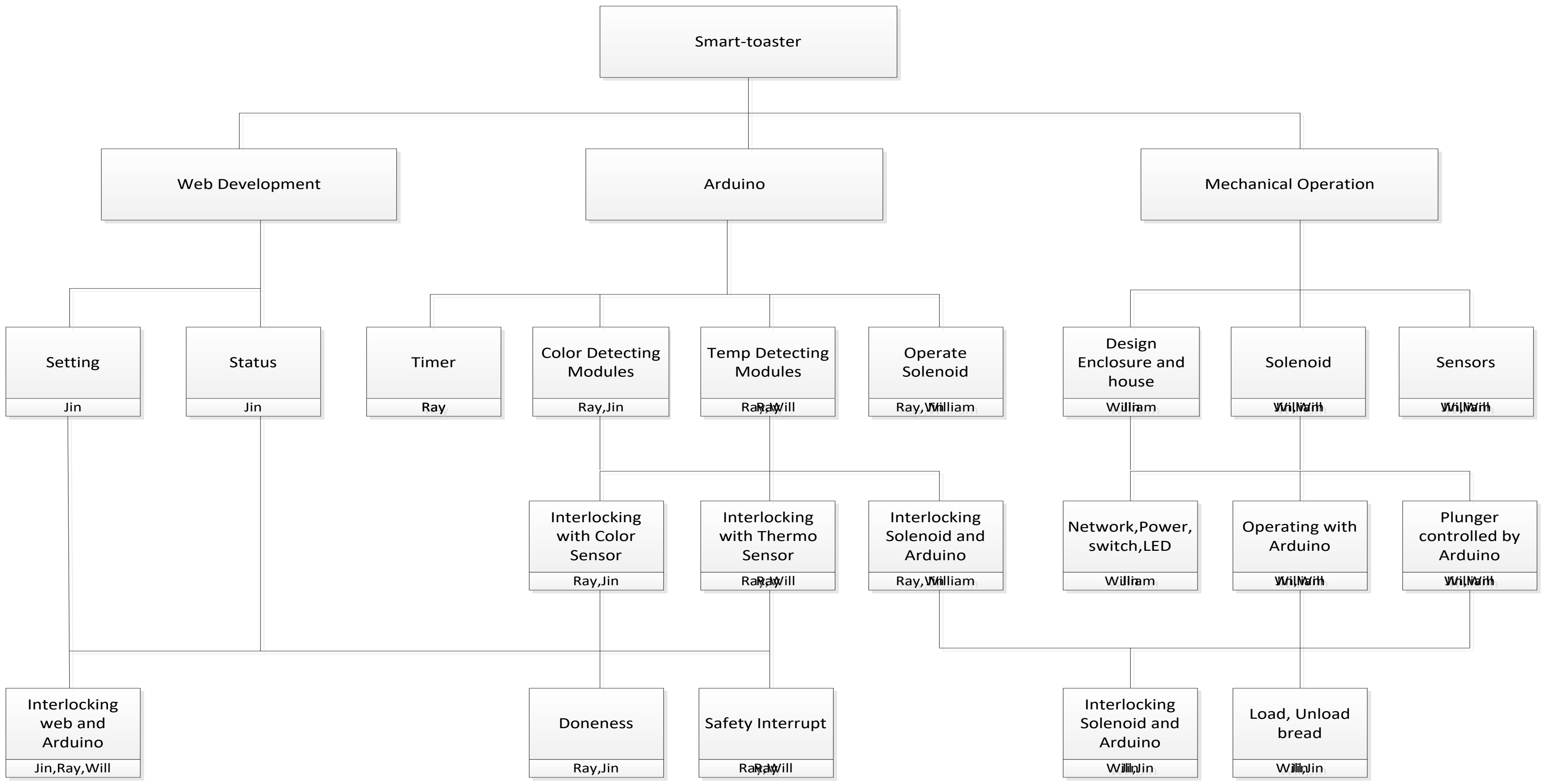
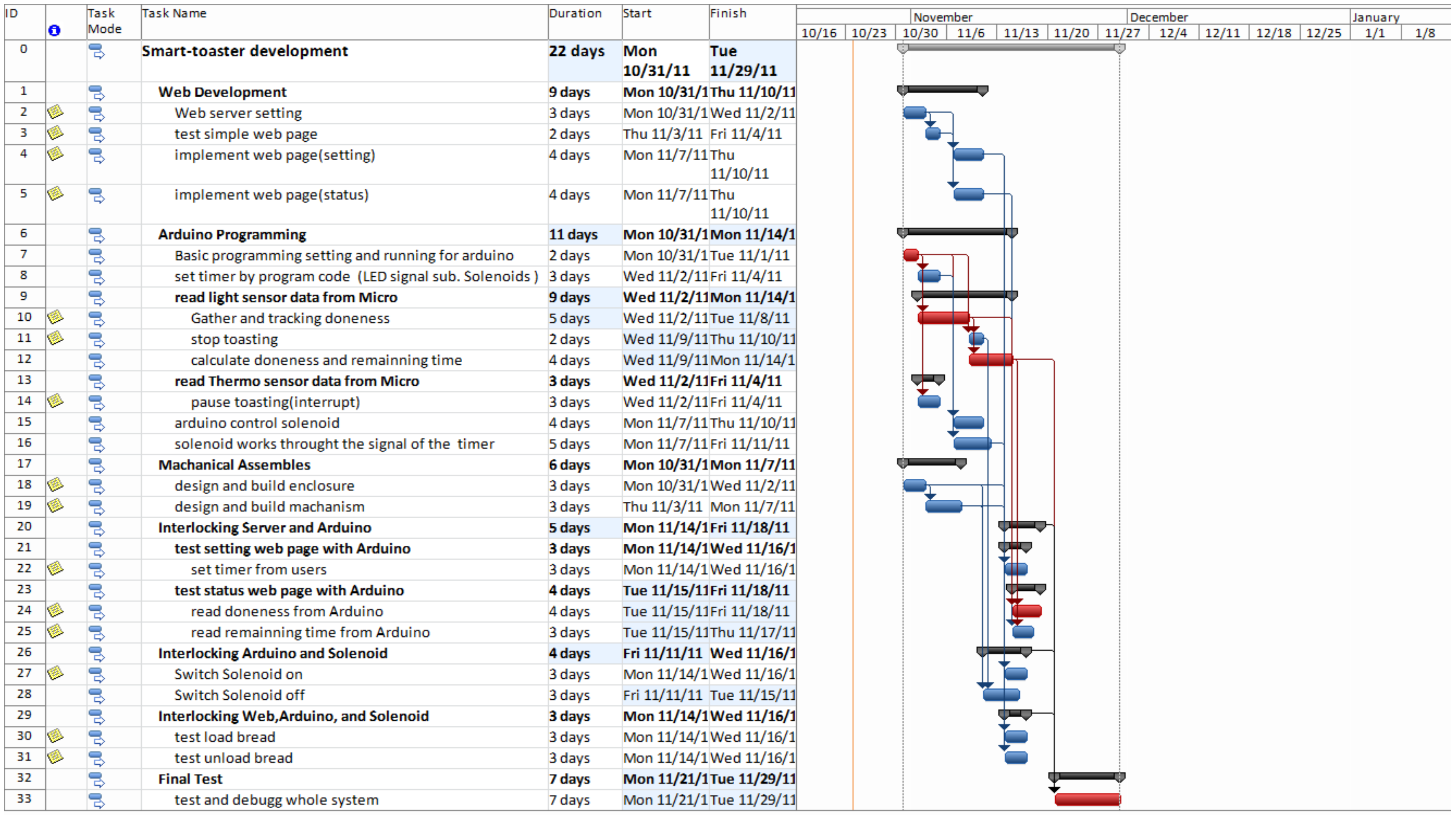


Figure xxxx

GANTT CHART



ID	ℹ	Task Mode	Task Name	Duration	Start	Finish
0			**Smart-toaster development**	**22 days**	**Mon 10/31/11**	**Tue 11/29/11**
1			**Web Development**	**9 days**	**Mon 10/31/1**	**Thu 11/10/11**
2	📝		Web server setting	3 days	Mon 10/31/1	Wed 11/2/11
3	📝		test simple web page	2 days	Thu 11/3/11	Fri 11/4/11
4	📝		implement web page(setting)	4 days	Mon 11/7/11	Thu 11/10/11
5	📝		implement web page(status)	4 days	Mon 11/7/11	Thu 11/10/11
6			**Arduino Programming**	**11 days**	**Mon 10/31/1**	**Mon 11/14/1**
7			Basic programming setting and running for arduino	2 days	Mon 10/31/1	Tue 11/1/11
8			set timer by program code (LED signal sub. Solenoids)	3 days	Wed 11/2/11	Fri 11/4/11
9			**read light sensor data from Micro**	**9 days**	**Wed 11/2/11**	**Mon 11/14/1**
10	📝		Gather and tracking doneness	5 days	Wed 11/2/11	Tue 11/8/11
11	📝		stop toasting	2 days	Wed 11/9/11	Thu 11/10/11
12			calculate doneness and remainning time	4 days	Wed 11/9/11	Mon 11/14/1
13			**read Thermo sensor data from Micro**	**3 days**	**Wed 11/2/11**	**Fri 11/4/11**
14	📝		pause toasting(interrupt)	3 days	Wed 11/2/11	Fri 11/4/11
15			arduino control solenoid	4 days	Mon 11/7/11	Thu 11/10/11
16			solenoid works throught the signal of the timer	5 days	Mon 11/7/11	Fri 11/11/11
17			**Machanical Assembles**	**6 days**	**Mon 10/31/1**	**Mon 11/7/11**
18	📝		design and build enclosure	3 days	Mon 10/31/1	Wed 11/2/11
19	📝		design and build machanism	3 days	Thu 11/3/11	Mon 11/7/11
20			**Interlocking Server and Arduino**	**5 days**	**Mon 11/14/1**	**Fri 11/18/11**
21			**test setting web page with Arduino**	**3 days**	**Mon 11/14/1**	**Wed 11/16/1**
22	📝		set timer from users	3 days	Mon 11/14/1	Wed 11/16/1
23			**test status web page with Arduino**	**4 days**	**Tue 11/15/11**	**Fri 11/18/11**
24	📝		read doneness from Arduino	4 days	Tue 11/15/11	Fri 11/18/11
25	📝		read remainning time from Arduino	3 days	Tue 11/15/11	Thu 11/17/11
26			**Interlocking Arduino and Solenoid**	**4 days**	**Fri 11/11/11**	**Wed 11/16/1**
27	📝		Switch Solenoid on	3 days	Mon 11/14/1	Wed 11/16/1
28			Switch Solenoid off	3 days	Fri 11/11/11	Tue 11/15/11
29			**Interlocking Web,Arduino, and Solenoid**	**3 days**	**Mon 11/14/1**	**Wed 11/16/1**
30	📝		test load bread	3 days	Mon 11/14/1	Wed 11/16/1
31	📝		test unload bread	3 days	Mon 11/14/1	Wed 11/16/1
32			**Final Test**	**7 days**	**Mon 11/21/1**	**Tue 11/29/11**
33			test and debugg whole system	7 days	Mon 11/21/1	Tue 11/29/11

Figure xxxx

STAKEHOLDERS AND THEIR RESPONSIBILITIES

Jin Won Seo – Engineering Team

William McLachlan – Engineering Team

Raymond Chan – Engineering team

Ron Wlock – Project Management Instructor

Robert Trost – Project Course Instructor

The engineering team members are responsible for updating the documents with timely and accurate data. Keeping track of progress of the projects is a key objective of the project, as it allows proper tracking of progress that will allow analysis of successes and timeliness of completion. The team members are also responsible for completing the Smart-Toaster® project on time and according to the specifications. The work breakdown structure outlined by the team members defines the individual responsibilities per each task they have been assigned to complete for the group. Failure of any member to complete their tasks to a satisfactory degree could put unnecessary stress on the other team members, completing the project is of vast importance to every team member as their graduation statuses rely on a well done technical project and report.

The Projects course instructors have interest in the project and in weekly updates on its status and progress.

MEASURE OF SUCCESS

Objective Functions

The toaster would be considered as a functionally successful product if following tasks can function well.

1. The toaster will be controller by the micro-controller to toast to the exact level of darkness that was specified by the user over the webpage interface.
2. Through network connection between the toaster and webpage interface, users can observe the toasting progress level of the bread, setting date and the remaining time.
3. Through the webpage interface, User can select timing mode in which users can set the start time of toasting and select the toasted level mode in which users can adjust their desire doneness and have their toast made the way they want when they want.
4. User can click the start button on the GUI interface to turn on the toaster right away.
5. In timing mode, the toaster starts cooking at the time specified by the user.
6. In toasted level mode, the toaster processes the job based on the doneness of the toasting level, in other word, the toaster will turn off after the light sensor signal detect the darkness excess particular value that is set by user setting through the webpage interface.

DELIVERABLES

Features

A smart function special toaster includes the following features:

- The status shows the process remain time and the doneness of the toasting show on the GUI web interface.
- The timer mode initializes the toaster at the setting time, and turns off when toast is ready.
- The toasting level mode will do the toasting until the darkness of the bread reaches the setting point.
- Shut down the toaster if temperature excess a particular dangerous value

Documentation

The design documents consist of the following:

-Schematics
-Flowchart
-Block diagram
-Mechanical drawing

Timeline
-Gantt chart
-Division of labor

Review

The report will tell the research, the development in hardware and programming, and all the test results during the project period for the entire smart toaster project.

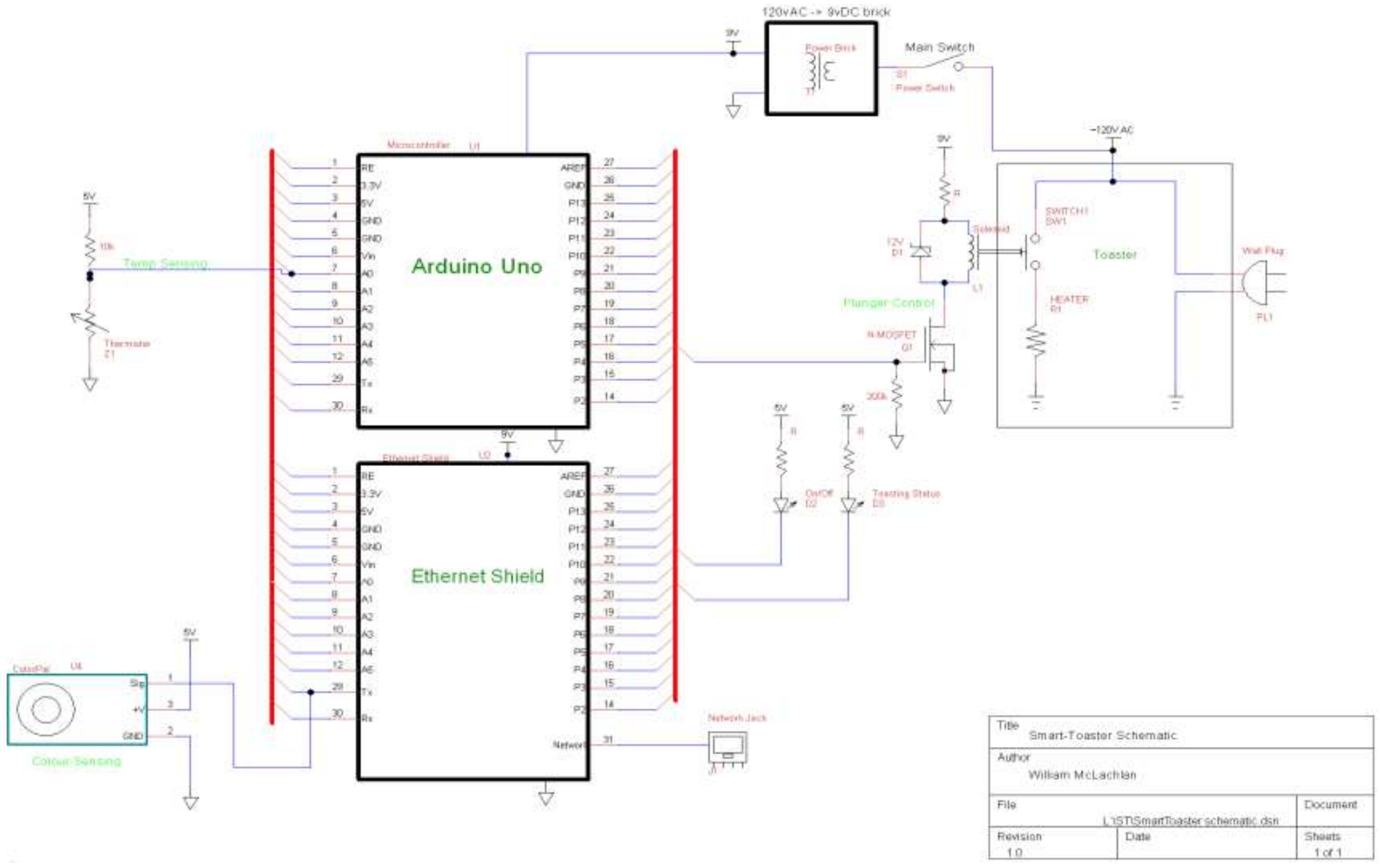

120vAC -> 9vDC brick
Power Brick
Main Switch
Power Switch
~120V AC
9V
5V
10k
Temp Sensing
Thermistor
Microcontroller U1
Arduino Uno
Ethernet Shield
Solenoid
12V D1
Plunger Control
N-MOSFET Q1
300k
SWITCH1 SW1
HEATER R1
Toaster
Wall Plug
On/Off D2
Toasting Status D3
ColorPal U4
Colour Sensing
Network Jack
Title Smart-Toaster Schematic
Author William McLachlan
File L:\ST\SmartToaster schematic.dsn
Document
Revision 1.0
Date
Sheets 1 of 1

Figure xxxx

SCHEMATIC

- Pins on Arduino and the Ethernet Shield are connected by stacked headers. Indicated by the bus lines on the schematic is that each pin on the Arduino is the same pin located on the Ethernet shield.
- The Plunger control solenoid circuit's Rail voltage (shown as 9V) and current limit resistor will be spec'd out according to the solenoid to be bought. Solenoid not yet sourced, the voltage and current levels needed are not yet known. The controlling FET will also need to meet the solenoid's current level.
- Temperature sensing to be done by feeding the voltage on a temperature sensitive thermistor into the Arduino's ADC. The values read will correspond to temperatures in a predictable manner.
- The Colour sensing will be done by a Parallax ColorPal connected on a single open-collector serial line. All reads and writes to/from the colour sensor will be done over this line.
- Power is supplied to the Micro and etc by a salvaged AC->DC adaptor, outputting 9V. The Arduino board has 5V and 3.3V converters on it usable by outside peripherals (Temp and colour sensor for example).
- The Arduino output pins can sink up to 40mA, which is more than enough for the indicator LEDs D2 & D3. The current limit resistors for the LEDs will be set for a reasonable brightness. Positive voltage for the LED's will be provided by the +5V output pin. The data sheet for the Arduino states *"the total current sourced from all I/O pins must not exceed 200mA"* so minimal sourcing will be aimed for.