# CHICKEN COOP GANTT CHART

## Project Title
Smart Chicken Coop

## Project Managers
Mark Huynh, Nick McBee, Michael Crockett

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Week 8</th>
<th>Week 9</th>
<th>Week 10</th>
</tr>
</thead>
</table>

### Project Design and Research Phase 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Diagram</td>
<td>All</td>
<td>1/13/20</td>
</tr>
<tr>
<td>Gantt Chart Draft</td>
<td>All</td>
<td>1/13/20</td>
</tr>
<tr>
<td>Delegate Blocks to each person</td>
<td>All</td>
<td>1/13/20</td>
</tr>
<tr>
<td>Data Logger/Transmitter Block</td>
<td>Mark</td>
<td>1/14/21</td>
</tr>
<tr>
<td>Sensors and Decoder Block</td>
<td>Nick</td>
<td>1/14/21</td>
</tr>
<tr>
<td>Door and Motor Block</td>
<td>Michael</td>
<td>1/14/21</td>
</tr>
</tbody>
</table>

### Building and Testing Phase 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Logger/Transmitter Block</td>
<td>Mark</td>
<td>1/21/20</td>
</tr>
<tr>
<td>Build Door and Motor Block</td>
<td>Michael</td>
<td>1/21/20</td>
</tr>
<tr>
<td>Build Sensors and Decoder Block</td>
<td>Nick</td>
<td>1/21/20</td>
</tr>
</tbody>
</table>

### Project Design and Research Phase 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning Light Controller Research</td>
<td>Mark</td>
<td>2/1/21</td>
</tr>
<tr>
<td>Heat Lamp Dimmer Block</td>
<td>Nick</td>
<td>2/1/21</td>
</tr>
<tr>
<td>Enclosure Research</td>
<td>Michael</td>
<td>2/1/21</td>
</tr>
<tr>
<td>Design Feedback Session</td>
<td>All</td>
<td>2/3/21</td>
</tr>
</tbody>
</table>

### Building and Testing Phase 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Warning Light Controller Block</td>
<td>Mark</td>
<td>2/9/21</td>
</tr>
<tr>
<td>Build Enclosure Block</td>
<td>Michael</td>
<td>2/9/21</td>
</tr>
<tr>
<td>Build Heat Lamp Dimmer Research</td>
<td>Nick</td>
<td>2/9/21</td>
</tr>
</tbody>
</table>

### Second Block Checkoff

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second Block Checkoff</td>
<td>All</td>
<td>2/19/21</td>
</tr>
</tbody>
</table>

### Project Presentation

<table>
<thead>
<tr>
<th>Task</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Verification</td>
<td>All</td>
<td>2/20/21</td>
</tr>
<tr>
<td>Project Showcase</td>
<td>All</td>
<td>2/20/21</td>
</tr>
<tr>
<td>Team Video</td>
<td>All</td>
<td>3/6/21</td>
</tr>
<tr>
<td>End of Project Team Assessment</td>
<td>All</td>
<td>3/6/21</td>
</tr>
<tr>
<td>Part Name</td>
<td>Quantity</td>
<td>Price</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>Data Transmitter Block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arduino Nano 33 IoT</td>
<td>1</td>
<td>$18</td>
</tr>
<tr>
<td>Door Motor Block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motor</td>
<td>1</td>
<td>$8</td>
</tr>
<tr>
<td>Jumper Wires</td>
<td>7</td>
<td>Owned</td>
</tr>
<tr>
<td>Door (Cardboard)</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>Doorframe (Cardboard)</td>
<td>1</td>
<td>Owned</td>
</tr>
<tr>
<td>L293 Transistor</td>
<td>1</td>
<td>Owned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Module Block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thermistor</td>
<td>1</td>
<td>$1</td>
</tr>
<tr>
<td>RTC Module</td>
<td>1</td>
<td>$2.20</td>
</tr>
<tr>
<td>Photoresistor</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1/4W or 1/8W 10k THT resistor</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>PCB</td>
<td>1</td>
<td>$4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warning Lights Block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Through Hole LEDs</td>
<td>1</td>
<td>$7.25</td>
</tr>
<tr>
<td>2 Channel Relays</td>
<td>2</td>
<td>$3</td>
</tr>
<tr>
<td>220 Ohm Resistors</td>
<td>4</td>
<td>$1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enclosure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Printed Model</td>
<td>1</td>
<td>$0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamp Dimmer Block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Photoresistor</td>
<td>1</td>
<td>$2</td>
</tr>
<tr>
<td>AC Triac dimmer board</td>
<td>1</td>
<td>$24</td>
</tr>
<tr>
<td>1/4W or 1/8W 10k THT resistor</td>
<td>2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Total Price $86.85
merged_coop.html

/*  merged_coop
*  Complete code used for system verification
*/

#include "thingProperties.h"
#define LAMP_MALFUNCTION_TRESH 1000 //Comparison value for ADC brightness measurement
#define LAMP_ON_AM 6 // Time that lamp will turn on in morning
#define LAMP_OFF_AM 9 // Time lamp will turn off in morning
#define LAMP_ON_PM 18 // Time lamp will turn on at night
#define LAMP_OFF_PM 21 // Time lamp will turn off at night
#define LAMP_WATTAGE 4 //Wattage rating of lamp bulb

#include "RTClib.h"
RTC_DS3231 rtc;
#include "NewPing.h"

const int dimmerPin = 6; //Pin for PWM control of dimmer board
const int lampPhotoPin = A3; //Photoresistor used for heat lamp
const int RED_PIN = 7; // the Arduino pin, which connects to the IN pin of relay for the RED LED
const int YELLOW_PIN = 8; // the Arduino pin, which connects to the IN pin of relay for the YELLOW LED
const int GREEN_PIN = 4; // the Arduino pin, which connects to the IN pin of relay for the GREEN LED
const int ALARM_PIN = A5; // the Arduino pin, which connects to the IN pin of relay for the BLUE LED

byte brightness_setting = 0; //Brightness setting for lamp
bool lamp_malfunction = 0; //Indicates malfunction when high

#define INPUT1 9 // Motor driver input 1
#define INPUT2 10 // Motor driver input 2
#define ENABLE 11 // Motor driver enable

#define Low 75 // determined as the optimal speed after testing
int Speed;

const byte photosensorPin = A0; // Environment light sensor pin
const byte thermistorPin = A1; //Steinhart-Hart constants for temperature calculations
const float c1 = 1.009249522e-03, c2 = 2.37845444e-04, c3 = 2.019202697e-07;
const int R3 = 10000; //Value of resistor in series with thermistor

const byte food_data_pin = 2;
const byte water_data_pin = 3;
const float foodDensity = 0.7; //grams per mL
const float containerLength = 20;
const float containerRadius = 4.0;
const float waterContLength = 16.0;
const float waterContRadius = 4.25;
const float waterDensity = 1.0;
const int samples = 5; //Number of ultrasonic readings to take per call

bool isOpen = false; // Current door state

NewPing food_sensor(food_data_pin, food_data_pin, 20);
NewPing water_sensor(water_data_pin, water_data_pin, 16);

float temp;
int currTime[3];
bool lightLevel;

float foodLevel, waterLevel;

void setup() {

  //Setup for transmitter module
  Serial.begin(57600);
  delay(1500); // Search for serial monitor
  //initProperties(); //Setup for transmitter
  Serial.println("Starting code");
  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  //Setup for sensors module
  pinMode(thermistorPin, INPUT);

  #ifndef ESP8266
    while (!Serial); // wait for serial port to connect. Needed for native USB
  #endif

  if (! rtc.begin()) {

Serial.println("Couldn't find RTC");
Serial.flush();
abort();

//Pin setup for door controller
pinMode(ENABLE, OUTPUT);
pinMode(INPUT1, OUTPUT);
pinMode(INPUT2, OUTPUT);

//Pin setup for warning lights
pinMode(RED_PIN, OUTPUT);
pinMode(YELLOW_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(ALARM_PIN, OUTPUT);
Serial.println("Leaving setup");

Speed = Low; // Set door motor speed

void loop() {
  //Update values and print to monitor
  getTime();
temp = getTemp();
lightlevel = getIfDaylight();
foodlevel = getFoodlevel();
waterlevel = getWaterLevel();
printData();

  //Update cloud variables/graphs
  //ArduinoCloud.update();
temperature = getTemp();
waterlevel = getWaterLevel();
foodlevel = getFoodlevel();
powerConsumption = get_power_consumption();
lightlevel = getIfDaylight();
lamp_controller(); //Handles heat lamp dimming and malfunction checking
print_lamp_data();
getTime();
printData(); //Print sensor values to serial monitor for testing

doorController(); //Handles door opening and closing
cchangeLights(waterLevel, foodLevel, temp, lamp_malfunction); //Handles warning light outputs
delay(1000);
}

/* Func: print_lamp_data
* Test bench function used to print the current test values
* for time and heat lamp status
*/
void print_lamp_data() {
  Serial.print("Power Consumption: ");
Serial.println(power_consumption);
if(lamp_malfunction)
   Serial.println("lamp malfunction detected");
else
   Serial.println("Lamp OK");
   Serial.println("\n");
}

/* Func: lamp_controller
* Determines if the heat lamp should be turned on based
* on current time and how bright it should be
*/
void lamp_controller()
if(brightness_setting == 255) //Checks lamp when fully turned on
   lamp_malfunction = check_lamp();
if((((currTime[0] >= LAMP_ON_AM) && (currTime[0] < LAMP_OFF_AM)) || temp <= 2) //Turns on lamp in early morning or in freezing temp
   brightness_setting = 255;
else if(currTime[0] >= LAMP_ON_PM) && (currTime[0] < LAMP_OFF_PM) //Turns on lamp in evening
   brightness_setting = 255;
else if(currTime[0] == LAMP_OFF_PM) //Dim lamp until off at nighttime
   dim_lamp();
else
   brightness_setting = 0; //Turn off lamp at all other times
}
float get_power_consumption() {
    float lamp_time = (LAMP_OFF_AM - LAMP_ON_AM) + (LAMP_OFF_PM - LAMP_ON_PM); //Time lamp is turned on
    float power = LAMP_WATTAGE * lamp_time; //Compute watt-hour usage
    return power;
}

/* Func: dim_lamp */
/* Lowers the lamp brightness with respect to number of seconds elapsed.
 * Should only be called when it is time for the lamp to be dimmed and turned off. */
void dim_lamp()
{
    if (brightness_setting <= 5) //Turns off lamp when dimming is complete
        brightness_setting = 0;
    else
        brightness_setting = 255 - 10 * currTime[2]; //Reduces brightness setting by about 4% each second
}

/* Func: check_lamp */
/* Reads the analog voltage from the lamp photocell divider to determine if
 * the lamp is functioning properly. This should only be called when the lamp is set at maximum brightness
 * to remain accurate. */
bool check_lamp()
{
    int lamp_brightness = analogRead(lampPhotoPin);
    if (lamp_brightness < LAMP_MALFUNCTION_TRESH) //Check if ADC reading is at corresponding light threshold
        return 1;
    return 0;
}

void Forward(void) //Starts the motor in a counterclockwise rotation in order to open the door
{
    analogWrite(ENABLE, Speed);
    digitalWrite(INPUT1, HIGH);
    digitalWrite(INPUT2, LOW);
}

void Backward(void) //Starts the motor in a counterclockwise rotation in order to close the door
{
    analogWrite(ENABLE, Speed);
    digitalWrite(INPUT1, LOW);
    digitalWrite(INPUT2, HIGH);
}

void stopped(void)//Function stops motor by setting both inputs to high
{
    digitalWrite(INPUT1, HIGH);
    digitalWrite(INPUT2, HIGH);
}

void opening(void) {
    Forward(); //opens door
    delay(300);
    stopped();
}

void closing(void) {
    Backward(); //closes door
    delay(300);
    stopped();
}

void doorController() {
    Speed = LOW; // Testing showed that 75 was about as slow as it could be set while still maintaining reliable operation.
    if ((currTime[0] == 19) && (currTime[1] == 50)) {
        opening();
        delay(300);
    }
    if ((currTime[0] == 5) && (currTime[1] == 0)) {
        closing();
    }
}

/* Function: getFoodLevel */
/* Determines the depth of the food in the container from
 * distance sensor and known constants. */
float getFoodLevel() {
    float h1 = get_food_distance(); //Distance of food from sensor
float h2 = containerLength - h1; //Depth of food in container
if(h2 < 0)
  h2 = 0;
float volume = PI*containerRadius*containerRadius*h2; //Food volume in cm^3
float mass = foodDensity*volume;
return mass;

//Water version of getFoodLevel
float getWaterLevel() {
  float h1 = get_water_distance(); //Distance of food from sensor
  float h2 = waterContLength - h1; //Depth of food in container
  if(h2 < 0)
    h2 = 0;
  float volume = PI*waterContRadius*waterContRadius*h2; //Food volume in cm^3
  float mass = waterDensity*volume;
  return mass;
}

/*Function: getFoodDistance
* Calculates the distance (in cm) of the food level from
* the food ultrasonic sensor
*/
float get_food_distance() {
  float duration, cm;
  duration = food_sensor.ping_median(samples);
  cm = (duration / 2) * 0.0343;
  return cm;
}

//Water version of get_food_distance
float get_water_distance() {
  float duration, cm;
  duration = water_sensor.ping_median(samples);
  cm = (duration / 2) * 0.0343;
  return cm;
}

/*Function: getIfDaylight
* Reads analog value from photoresistor voltage divider
* and provides a light level approximation.
*/
bool getIfDaylight() {
  int adcVal = analogRead(photosensorPin);
  float photoVo = 5*(float)adcVal/(1023); //Output voltage of voltage divider
  if(photoVo >= 4.17) { //Check if light level is at least 40 Lux
    return true;
  }
  return false;
}

/*Function: getTemp
*Reads analog value from thermistor voltage divider and calculates the temperature in Celsius.
*/
float getTemp() {
  int adcVal = analogRead(thermistorPin);
  float rTherm = R3*(1023.0/(float)adcVal - 1.0); //Current thermistor resistance
  float logRtherm = log(rTherm);
  //Temperature in Kelvin from Steinhart equation:
  float tempK = (1.0/(c1 + c2*logRtherm + c3*logRtherm*logRtherm*logRtherm));
  return (tempK - 273.15); //Return temp converted to Celsius
}

/*Function: printData
*Prints the sensor values to the
*serial monitor for testing and debugging.
*/
void printData() {
  Serial.print("Time: ");
  Serial.print(currTime[0]);
  Serial.print(":");
  Serial.print(currTime[1]);
  Serial.print(":");
  Serial.println(currTime[2]);
  Serial.print("Temperature (C): ");
  Serial.println(temp);
  if(lightLevel)
    Serial.println("Daylight brightness detected");
else
    Serial.println("Light level below daylight levels");

    Serial.print("Food Level: ");
    Serial.println(foodLevel);
    Serial.print("Water Level: ");
    Serial.println(waterLevel);
    Serial.println(" ");
}

/*Function: getTime
 *Collects current time info from
 *the RTC module and updates the
 *current time array.
 */
void getTime()
{
    DateTime now = rtc.now();
    currTime[0] = now.hour();
    currTime[1] = now.minute();
    currTime[2] = now.second();
}

void onFooChange()
{
    // These functions are intentionally left blank. They are what enable IoT cloud to cause changes based off of variable values
}

void onTemperatureChange()
{
    // Do something
}

void onFoodLevelChange()
{
    // Do something
}

void onWaterLevelChange()
{
    // Do something
}

void onLightLevelChange()
{
    // Do something
}

void onPowerConsumptionChange()
{
    // Do something
}

// Function that will take each of the sensor values and determine corresponding LEDs to turn on
void changeLights(float waterLvl, float foodLvl, float temperature, bool lampMalfunction)
{
    int yellow_light = waterLvl < 200 || foodLvl < 200; //Resources threshold
    int red_light = temperature <= 3; // Temperature threshold
    int green_light = !(yellow_light || red_light); // If no other lights are on, green should be on

    // Write all of the determined values to corresponding digital pins
    digitalWrite(RED_PIN, red_light);
    digitalWrite(YELLOW_PIN, yellow_light);
    digitalWrite(GREEN_PIN, green_light);
}

// Function to return lights back to green status
void returnNormalLights()
{
    changeLights(100, 100, 100, 100);
    delay(1000);
}
<table>
<thead>
<tr>
<th>Date</th>
<th>In</th>
<th>Out</th>
<th>Focus</th>
<th>Hours</th>
<th>Nick</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10</td>
<td>2:00 PM</td>
<td>3:00 PM</td>
<td>Requirements</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1/13</td>
<td>2:00 PM</td>
<td>4:00 PM</td>
<td>Block diagram</td>
<td>2</td>
<td>73</td>
</tr>
<tr>
<td>1/14</td>
<td>10:00 AM</td>
<td>12:00 PM</td>
<td>Block diagram</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1/15</td>
<td>3:00 PM</td>
<td>5:00 PM</td>
<td>Sensors research</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1/16</td>
<td>9:30 AM</td>
<td>3:00 PM</td>
<td>Sensor schematic</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>1/19</td>
<td>9:00 AM</td>
<td>1:00 PM</td>
<td>Sensor code</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1/21</td>
<td>11:00 AM</td>
<td>2:00 PM</td>
<td>Sensor testing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1/23</td>
<td>9:30 AM</td>
<td>3:00 PM</td>
<td>Sensor docs</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>1/26</td>
<td>10:00 AM</td>
<td>4:00 PM</td>
<td>Sensor testing</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1/29</td>
<td>3:00 PM</td>
<td>6:00 PM</td>
<td>Sensor docs</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2/2</td>
<td>4:00 PM</td>
<td>5:30 PM</td>
<td>Feedback docs</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>2/9</td>
<td>9:00 AM</td>
<td>1:00 PM</td>
<td>Dimmer research</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2/11</td>
<td>3:00 PM</td>
<td>6:00 PM</td>
<td>Dimmer design</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2/13</td>
<td>10:00 AM</td>
<td>2:00 PM</td>
<td>Dimmer schematic</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2/16</td>
<td>11:00 AM</td>
<td>2:00 PM</td>
<td>PCB design</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2/18</td>
<td>3:00 PM</td>
<td>6:00 PM</td>
<td>PCB design</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2/21</td>
<td>10:00 AM</td>
<td>4:00 PM</td>
<td>Sensor testing</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2/23</td>
<td>10:00 AM</td>
<td>12:00 PM</td>
<td>Dimmer testing</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2/25</td>
<td>11:00 AM</td>
<td>1:00 PM</td>
<td>Dimmer video</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2/27</td>
<td>10:00 AM</td>
<td>12:00 PM</td>
<td>Sensor fixing</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2/28</td>
<td>11:30 AM</td>
<td>8:00 PM</td>
<td>System integration</td>
<td>8.5</td>
<td></td>
</tr>
</tbody>
</table>

Time Spent: 73 hours
<table>
<thead>
<tr>
<th>Date</th>
<th>In</th>
<th>Out</th>
<th>Hours</th>
<th>Michael</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10</td>
<td>2:00 PM</td>
<td>3:00 PM</td>
<td>1:00:00</td>
<td></td>
</tr>
<tr>
<td>1/13</td>
<td>2:00 PM</td>
<td>4:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>1/14</td>
<td>1:00 PM</td>
<td>4:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>1/16</td>
<td>12:00 PM</td>
<td>6:00 PM</td>
<td>6:00:00</td>
<td></td>
</tr>
<tr>
<td>1/18</td>
<td>8:00 PM</td>
<td>11:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>1/20</td>
<td>3:00 PM</td>
<td>4:00 PM</td>
<td>1:00:00</td>
<td></td>
</tr>
<tr>
<td>1/23</td>
<td>10:00 AM</td>
<td>5:00 PM</td>
<td>7:00:00</td>
<td></td>
</tr>
<tr>
<td>1/25</td>
<td>7:00 PM</td>
<td>11:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>1/28</td>
<td>3:00 PM</td>
<td>4:00 PM</td>
<td>1:00:00</td>
<td></td>
</tr>
<tr>
<td>1/29</td>
<td>1:00 PM</td>
<td>1:30 PM</td>
<td>0:30:00</td>
<td></td>
</tr>
<tr>
<td>2/1</td>
<td>7:00 PM</td>
<td>10:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>2/4</td>
<td>2:00 PM</td>
<td>6:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>2/6</td>
<td>11:00 AM</td>
<td>2:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>2/11</td>
<td>11:00 AM</td>
<td>1:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>2/18</td>
<td>1:00 PM</td>
<td>5:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>2/19</td>
<td>11:30 AM</td>
<td>12:00 PM</td>
<td>0:30:00</td>
<td></td>
</tr>
<tr>
<td>2/25</td>
<td>1:00 PM</td>
<td>3:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>2/28</td>
<td>11:00 AM</td>
<td>7:00 PM</td>
<td>8:00:00</td>
<td></td>
</tr>
<tr>
<td>3/2</td>
<td>2:00 PM</td>
<td>8:00 PM</td>
<td>6:00:00</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>In</td>
<td>Out</td>
<td>Hours</td>
<td>Mark</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>---------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>1/11</td>
<td>6:00 PM</td>
<td>10:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>1/12</td>
<td>1:00 PM</td>
<td>3:00 PM</td>
<td>2:00:00</td>
<td>67:00:00</td>
</tr>
<tr>
<td>1/15</td>
<td>2:00 PM</td>
<td>5:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>1/17</td>
<td>5:00 PM</td>
<td>10:00 PM</td>
<td>5:00:00</td>
<td></td>
</tr>
<tr>
<td>1/20</td>
<td>12:00 PM</td>
<td>3:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>1/25</td>
<td>3:00 PM</td>
<td>4:00 PM</td>
<td>1:00:00</td>
<td></td>
</tr>
<tr>
<td>1/27</td>
<td>6:00 PM</td>
<td>11:00 PM</td>
<td>5:00:00</td>
<td></td>
</tr>
<tr>
<td>1/29</td>
<td>5:00 PM</td>
<td>7:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>1/30</td>
<td>2:00 PM</td>
<td>5:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>2/3</td>
<td>5:00 PM</td>
<td>8:00 PM</td>
<td>3:00:00</td>
<td></td>
</tr>
<tr>
<td>2/5</td>
<td>2:00 PM</td>
<td>6:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>2/7</td>
<td>11:00 AM</td>
<td>1:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>2/11</td>
<td>11:00 AM</td>
<td>1:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>2/18</td>
<td>1:00 PM</td>
<td>5:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>2/19</td>
<td>11:00 AM</td>
<td>4:00 PM</td>
<td>5:00:00</td>
<td></td>
</tr>
<tr>
<td>2/21</td>
<td>11:00 AM</td>
<td>3:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>2/28</td>
<td>11:00 AM</td>
<td>8:00 PM</td>
<td>9:00:00</td>
<td></td>
</tr>
<tr>
<td>3/2</td>
<td>7:00 PM</td>
<td>9:00 PM</td>
<td>2:00:00</td>
<td></td>
</tr>
<tr>
<td>3/3</td>
<td>11:00 AM</td>
<td>3:00 PM</td>
<td>4:00:00</td>
<td></td>
</tr>
<tr>
<td>Team Member</td>
<td>Time Spent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nick</td>
<td>73:00:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td>61:00:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mark</td>
<td>67:00:00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Time Spent**

- **Mark**: 33.3%
- **Nick**: 36.3%
- **Michael**: 30.3%