1.

**** main.c ****

```
#include "gpio.h"
#include "wwv.h"

#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>

// Run tests on gpio.c functions
int main(int argc, char *argv[])
{
    uint32_t retvalue;

    retvalue = encodedatetime(4);
    if (retvalue != 0) {
        printf("Error encoding!\n");
        return -1;
    }

    return 0;
}
```

**** wwv.h ****

```
#ifndef WWV_H
#define WWV_H

// includes to be used
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#include <sys/resource.h>
#include "gpio.h"

// prototype functions
uint32_t encodebit(uint32_t gpio, uint32_t t_high);
uint32_t encodedatetime(uint32_t gpio);

#endif
```

**** wwv.c ****

```
#include "wwv.h"

// Encodes bit on gpio pin based on the time high passed
uint32_t encodebit(uint32_t gpio, uint32_t t_high)
{
    // timespec structs to get accurate timing
    struct timespec start;
    struct timespec finish;
    struct timespec start100hz;
    struct timespec finish100hz;

    // If time high is 0 just set output of gpio low for 1 sec
    if (t_high == 0) {
        clock_gettime(CLOCK_REALTIME, &start);
        if(gpio_setvalue(gpio,0) != 0) {
            printf("Unable to set gpio!\n");
        }
```

```
        while ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) - ((dou
ble)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) <= 1.0) {
            usleep(10);
            clock_gettime(CLOCK_REALTIME, &finish);
        }
        return 0;
    }

    // If time high is not 0, keep gpio high for that long then keep it low for the remaini
ng second
    // Multiple check for if time is over a second and if tim is above time high
    else {
        clock_gettime(CLOCK_REALTIME, &start);
        while ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) - ((dou
ble)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) <= ((double)t_high / 1000.0))
 {

            // Time high for the 100Hz signal
            clock_gettime(CLOCK_REALTIME, &start100hz);
            if(gpio_setvalue(gpio,1) != 0) {
                printf("Unable to set gpio!\n");
            }
            while ((((double)finish100hz.tv_sec + (double)finish100hz.tv_nsec/(double)10000
00000) - ((double)start100hz.tv_sec + (double)start100hz.tv_nsec/(double)1000000000)) <= 0.
005) {
                usleep(10);
                clock_gettime(CLOCK_REALTIME, &finish100hz);
                clock_gettime(CLOCK_REALTIME, &finish);
                if ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) -
((double)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) >= ((double)t_high / 100
0.0)) break;
            }
            if ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) - ((do
uble)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) >= ((double)t_high / 1000.0)
) break;
            // Time Low for the 100Hz signal
            clock_gettime(CLOCK_REALTIME, &start100hz);
            if(gpio_setvalue(gpio,0) != 0) {
                printf("Unable to set gpio!\n");
            }
            while ((((double)finish100hz.tv_sec + (double)finish100hz.tv_nsec/(double)10000
00000) - ((double)start100hz.tv_sec + (double)start100hz.tv_nsec/(double)1000000000)) <= 0.
005) {
                usleep(10);
                clock_gettime(CLOCK_REALTIME, &finish100hz);
                clock_gettime(CLOCK_REALTIME, &finish);
                if ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) -
((double)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) >= ((double)t_high / 100
0.0)) break;
            }
        }
        if(gpio_setvalue(gpio,0) != 0) {
            printf("Unable to set gpio!\n");
        }
        while ((((double)finish.tv_sec + (double)finish.tv_nsec/(double)1000000000) - ((dou
ble)start.tv_sec + (double)start.tv_nsec/(double)1000000000)) <= 1.0) {
            usleep(10);
            clock_gettime(CLOCK_REALTIME, &finish);
        }

        return 0;
    }
}

// Gets data and time and encodes it for wwv
uint32_t encodedatetime(uint32_t gpio)
{
    uint32_t i;                 // Iterator
```

```
    uint32_t retvalue;          // Return value check
    uint32_t array[60] = {0};   // Hold bits for wwv
    uint32_t yearones;          // Date and Time variables
    uint32_t minones;
    uint32_t mintens;
    uint32_t hourones;
    uint32_t hourtens;
    uint32_t doyones;
    uint32_t doytens;
    uint32_t doyhundreds;
    time_t now;
    struct tm *local;

    // Exports and sets gpio
    gpio_export(gpio);
    gpio_setdirection(gpio, "out");

    // Gets date and time
    time(&now);
    local = localtime(&now);

    // Gets ones, tens and hundreds place for date and time
    yearones = (local->tm_year + 1900) % 10;
    minones = (local->tm_min) % 10;
    mintens = ((local->tm_min) % 100) / 10;
    hourones = (local->tm_hour) % 10;
    hourtens = ((local->tm_hour) % 100) / 10;
    doyones = (local->tm_yday) % 10;
    doytens = ((local->tm_yday) % 100) / 10;
    doyhundreds = ((local->tm_yday) % 1000) / 100;

    // Places 1s, 2s, 4s, and 8s bit in wwv array
    for (i = 4; i < 8; i++) {
        array[i] = ((yearones & ((i - 4) << 0x01)) ? 470 : 170);
        array[i + 6] = ((minones & ((i - 4) << 0x01)) ? 470 : 170);
        array[i + 16] = ((hourones & ((i - 4) << 0x01)) ? 470 : 170);
        array[i + 26] = ((doyones & ((i - 4) << 0x01)) ? 470 : 170);
        array[i + 31] = ((doytens & ((i - 4) << 0x01)) ? 470 : 170);
    }

    // Places 1s, 2s, and 4s bit in wwv array
    for (i = 15; i < 18; i++) {
        array[i] = ((mintens & ((i - 15) << 0x01)) ? 470 : 170);
        array[i + 10] = ((hourtens & ((i - 15) << 0x01)) ? 470 : 170);
    }

    // Places 1s, and 2s bit in wwv array
    for (i = 40; i < 42; i++) {
        array[i] = ((doyhundreds & ((i - 40) << 0x01)) ? 470 : 170);
    }

    // Places P indentifiers in wwv array
    for (i = 1; i < 5; i++) {
        array[i * 10 - 1] = 770;
    }

    // Prints time and date for checking purposes
    printf("Beginning to encode Year: %d, DoY: %d, Time: %d:%d", local->tm_year + 1900, loc
al->tm_yday, local->tm_hour, local->tm_min);

    // Prints and encodes bits for wwv
    for (i = 0; i < 60; i++) {
        if ((i % 10) == 0) {
            if (i == 50) {
                printf("\nEncoding Segment 0\n");
            }
            else {
                printf("\nEncoding Segment %d\n", (i / 10) + 1);
```

```
            }
        }

        retvalue = encodebit(gpio, array[i]);
        if (retvalue != 0) {
            printf("Couldn't Encode Time Index %d!\n", i);
            return -1;
        }
        else {
            if (array[i] == 770) {
                printf("Time Index %d: P\n", i);
            }
            else if (array[i] == 0) {
                printf("Time Index %d: 0\n", i);
            }
            else {
                printf("Time Index %d: %d\n", i, ((array[i] == 470) ? 1 : 0));
            }
        }
    }

    // Unexports gpio
    gpio_unexport(gpio);
    return 0;
}
```

**** gpio.h ****

```
#ifndef GPIO_H
#define GPIO_H

// includes to be used
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

// gpio function prototypes
uint32_t gpio_export(uint32_t gpio);
uint32_t gpio_unexport(uint32_t gpio);
uint32_t gpio_setvalue(uint32_t gpio, uint32_t val);
uint32_t gpio_setdirection(uint32_t gpio, char dir[]);

#endif
```

**** gpio.c ****

```
#include "gpio.h"

// Export gpio
uint32_t gpio_export(uint32_t gpio)
{
    int32_t f;  // file
    char string[10];    // write
    sprintf(string,"%u",gpio);  // change gpio from int to string

    // open export file
    f = open("/sys/class/gpio/export",O_WRONLY);
    if (f < 0) {
        printf("Could not open export\n");
        return 1;
    }

    // export gpio
    if (write(f,string,strlen(string)) < 0) {
```

```
        printf("Could not export\n");
        return 2;
    }

    // close and wait
    close(f);
    usleep(50000);

    return 0;
}

// Unexport gpio
uint32_t gpio_unexport(uint32_t gpio)
{
    int32_t f;  // file
    char string[10];    // write
    sprintf(string,"%u",gpio);  // change gpio from int to string

    // open unexport file
    f = open("/sys/class/gpio/unexport",O_WRONLY);
    if (f < 0) {
        printf("Could not open unexport\n");
        return 1;
    }

    // unexport gpio
    if(write(f,string,strlen(string)) < 0) {
        printf("Could not unexport\n");
        return 2;
    }

    // close and wait
    close(f);
    usleep(50000);

    return 0;
}

// Set value for gpio
uint32_t gpio_setvalue(uint32_t gpio, uint32_t val)
{
    int32_t f;  // file
    char string[50];    // write
    char string1[10];   // value
    sprintf(string,"/sys/class/gpio/gpio%u/value",gpio);    // change gpio from int to stri
ng
    sprintf(string1,"%u",val);  // change val from int to string

    // open value file
    f = open(string,O_WRONLY);
    if (f < 0) {
        printf("Could not open value\n");
        return 1;
    }

    // write value
    if(write(f,string1,strlen(string1)) < 0) {
        printf("Could not set value\n");
        return 2;
    }

    // close and wait
    close(f);
    usleep(50);

    return 0;
}
```

```
// Set direction for gpio
uint32_t gpio_setdirection(uint32_t gpio, char dir[])
{
    int32_t f;  // file
    char string[50];    // write
    sprintf(string,"/sys/class/gpio/gpio%u/direction",gpio);    // change gpio from int to
string

    // open direction file
    f = open(string,O_WRONLY);
    if (f < 0) {
        printf("Could not open direction\n");
        return 1;
    }

    // write value
    if (write(f,dir,strlen(dir)) < 0) {
        printf("Could not set direction\n");
        return 2;
    }

    // close and wait
    close(f);
    usleep(50);

    return 0;
}

2.
a)  sudo apt-get update
    sudo apt-get upgrade

b)  sudo git clone https://github.com/raspberrypi/linux.git

c)  sudo git log --grep=4.19.97
    sudo git checkout dc4ba5be1babd3b3ec905751a30df89a5899a7a9

d)  sudo mv linux linux4.19.97-v7l+

e)  make mrproper
    sudo modprobe configs
    sudo cp /proc/config.gz /usr/src/linux4.19.97-v7l+/config.gz
    sudo gunzip -dk config.gz
    sudo mv config .config
    sudo wget raw.githubusercontent.com/raspberrypi/firmware/01ecfd2ba2b7cf3a2f4aa75ada895e
e4a3e729f5/extra/Module7l.symvers
    sudo mv Module7l.symvers /boot/Module7l.symvers
    sudo ln -s /boot/Module7l.symvers Module.symvers
    sudo apt-get install bison
    sudo apt-get install flex
    sudo apt-get install bc
    sudo make modules_prepare

f)  sudo ln -s /usr/src/linux4.19.97-v7l+ /lib/modules/4.19.97-v7l+/build

g)  sudo ln /lib/modules/4.19.97-v7l+/build /lib/modules/4.19.97-v7l+/source

5. enscript -T 4 --header='$n %E %*|$%|Spencer Goulette' hw05_2.txt -o - | ps2pdf - output.
pdf
```