```
/******
  Programm for booking av rom:
  ----------------------------
  Programmet kontrollerer bookingen av bad, kjøkken og TV-stue i et kollektiv.
  Det henter inn verdier fra signaler fra nettsiden www.circusofthings.com.
  Verdiene signalene holder angir antall minutter et rom vil bookes. Programmet
  sjekker først om det er ledig plass for bookingen som blir gjort. Hvis det er
  ledig plass starter programmet en klokke som teller ned for det gitte antallet
  minutter. Programmet oppdaterer hvor lang tid det er igjen av bookingen tilbake
  til CoT. Dette skjer hvert 10. sekund. I tillegg til dette holder
  programmet også oversikt over hvor lenge TV-en i TV-stua har stått på. Dette
  med antagelsen at hver gang TV-stua blir booket er TV-en i bruk.
******/

// Importerer nødvendige biblioteker.
#include <CircusESP32Lib.h>

// Definerer faktor for omgjøring fra minutter til millisekunder
#define MIN_TIL_mS 60000

// Henter inn nødvendige nettverksdetaljer.
char ssid[] = "ssid";
char password[] = "password";

// Henter inn nødvendig informasjon som CoT og signaler fra nettsiden.
char server[] = "www.circusofthings.com";
char token[] = "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI1MTc3In0.G9gRducsNPjd8I01Pn6tmKB6hDr8MOXLr_t9cW
YNwYY";
char key_booking_BB[] = "15968";
char key_booking_KB[] = "32242";
char key_booking_LB[] = "16309";
char key_BB[] = "305";
char key_KB1[] = "3234";
char key_KB2[] = "1010";
char key_LB1[] = "16632";
char key_LB2[] = "28394";
char key_LB3[] = "21919";
char key_TV[] = "22181";

// Definerer nødvendige variabler for behandling av bookinger.
// [0] = Tiden en booking først blir registrert (millis()).
// [1] = Lengden bookingen skal vare.
// [2] = Om bookingplassen er i bruk eller ikke (1 eller 0).
unsigned long BB[] = {0, 0, 0};
unsigned long KB1[] = {0, 0, 0};
unsigned long KB2[] = {0, 0, 0};
unsigned long LB1[] = {0, 0, 0};
unsigned long LB2[] = {0, 0, 0};
unsigned long LB3[] = {0, 0, 0};

// Definerer variabler for oppdatering til CoT.
unsigned long update_time_CoT = 0;
unsigned long update_intervall_CoT = 10000;
```

```
// Definerer variabler for oppdatering av bruk av TV (+ bruk av TV i minutter).
unsigned long update_time_TV = 0;
unsigned long update_intervall_TV = 60000;
int TV_time_used = 0;

// Definerer integrer.
int booking_time_BB = 0;
int booking_time_KB = 0;
int booking_time_LB = 0;

// Definnerer integrer for pinner til fysiske tilkoblinger.
int GreenLED = 26;
int RedLED = 14;
int buzzer_LED = 32;

CircusESP32Lib circusESP32(server, ssid, password);

// SETUP
void setup() {
  circusESP32.begin();
  pinMode(GreenLED, OUTPUT);
  pinMode(RedLED, OUTPUT);
  pinMode(buzzer_LED, OUTPUT);
}

// LOOP
void loop() {

  // Sjekker om det er tid å oppdatere og hente signaler fra CoT.
  if (update_intervall_CoT <= (millis() - update_time_CoT)){
    update_time_CoT = millis();

    // Henter inn verdier fra booking-signaler fra CoT.
    // BADET
    booking_time_BB = circusESP32.read(key_booking_BB, token);
    // KJØKKENET
    booking_time_KB = circusESP32.read(key_booking_KB, token);
    // TV-STUA
    booking_time_LB = circusESP32.read(key_booking_LB, token);

    // Henter inn verdier fra TV-signalet i CoT.
    // Hvor lenge TV-en har hvert i bruk hentes inn i tilfelle RPien har
    // lest av verdien for dagens TV bruk fra CoT.
    // RPien setter da denne verdien til 0 for å starte tellingen for en ny dag.
    TV_time_used = circusESP32.read(key_TV, token);

    // Oppdateres signalene i CoT som viser hvor lang tid det er igjen av en booking til CoT
    // Dette gjelder kun for de signalene som er i bruk (signal[1] != 0).
    // BADET
    if (BB[1] != 0){
      circusESP32.write(key_BB, float(BB[1] + BB[0] - millis())/MIN_TIL_mS, token);
    }
    // KJØKKENET
    if (KB1[1] != 0){
```

```
    circusESP32.write(key_KB1, float(KB1[1] + KB1[0] - millis())/MIN_TIL_mS, token);
  }
  if (KB2[1] != 0){
    circusESP32.write(key_KB2, float(KB2[1] + KB2[0] - millis())/MIN_TIL_mS, token);
  }
  // TV-STUA
  if (LB1[1] != 0){
    circusESP32.write(key_LB1, float(LB1[1] + LB1[0] - millis())/MIN_TIL_mS, token);
  }
  if (LB2[1] != 0){
    circusESP32.write(key_LB2, float(LB2[1] + LB2[0] - millis())/MIN_TIL_mS, token);
  }
  if (LB3[1] != 0){
    circusESP32.write(key_LB3, float(LB3[1] + LB3[0] - millis())/MIN_TIL_mS, token);
  }
}
//---------------------------------------------------------------


// Sjekker om booking_time_** er ulik "0" (noen har gjort en booking fra CoT).
// Hvis en booking er gjort for et rom, sjekkes det om rommet har en plass som er ledig.
// Hvis det er en ledig plass på rommet blir bookingen registrert, hvis ikke blir den avvist.
// BADET
if (booking_time_BB != 0){
  if (BB[2] == 0){
    bookingGodkjent(BB, booking_time_BB);
  }
  else{
    Indikator(RedLED);
  }
  circusESP32.write(key_booking_BB, 0, token);
}
// KJØKKENET
if (booking_time_KB != 0){
  if (KB1[2] == 0){
    bookingGodkjent(KB1, booking_time_KB);
  }
  else if (KB2[2] == 0){
    bookingGodkjent(KB2, booking_time_KB);
  }
  else{
    Indikator(RedLED);
  }
  circusESP32.write(key_booking_KB, 0, token);
}
// TV-STUA
if (booking_time_LB != 0){
  if (LB1[2] == 0){
    bookingGodkjent(LB1, booking_time_LB);
  }
  else if (LB2[2] == 0){
    bookingGodkjent(LB2, booking_time_LB);
  }
  else if (LB3[2] == 0){
```

```
    bookingGodkjent(LB3, booking_time_LB);
   }
   else{
    Indikator(RedLED);
   }
   circusESP32.write(key_booking_LB, 0, token);
  }
  //------------------------------------------------------------


  // Sjekker om tiden er utgått for hver av bookingene som er gjort.
  // Hvis tiden er ute for en booking blir funksjonen bookingTidUtgaatt kalt.
  // BADET
  if ((BB[1] <= millis() - BB[0]) && (BB[2] == 1)){
   bookingTidUtaatt(BB, key_BB);
  }
  // KJØKKENET
  if ((KB1[1] <= millis() - KB1[0]) && (KB1[2] == 1)){
   bookingTidUtaatt(KB1, key_KB1);
  }
  if ((KB2[1] <= millis() - KB2[0]) && (KB2[2] == 1)){
   bookingTidUtaatt(KB2, key_KB2);
  }
  // TV-STUA
  if ((LB1[1] <= millis() - LB1[0]) && (LB1[2] == 1)){
   bookingTidUtaatt(LB1, key_LB1);
  }
  if ((LB2[1] <= millis() - LB2[0]) && (LB2[2] == 1)){
   bookingTidUtaatt(LB2, key_LB2);
  }
  if ((LB3[1] <= millis() - LB3[0]) && (LB3[2] == 1)){
   bookingTidUtaatt(LB3, key_LB3);
  }

  //------------------------------------------------------------


  // Sjekker om TV-en er i bruk.
  // Dette skjer hvert minutt.
  // Hvis TV-en er i bruk blir TV_timed_used inkrementert og oppdatert til CoT.
  if (update_intervall_TV <= (millis() - update_time_TV)){
   update_time_TV = millis();
   if ((LB1[2] == 1) || (LB2[2] == 1) || (LB3[2] == 1)){
    TV_time_used++;
    circusESP32.write(key_TV, TV_time_used, token);
   }
  }
  //------------------------------------------------------------
}

// FUNKSJONER
/******
  Kalles når en booking er godkjent. Tar inn to inputs og gir ingen outputs.
  Input-ene er booking nummeret bookingen skal registrers på og hvor lang tid
```

```
  bookingen skal vare. Leddene i booking nummerets array tilegnes sine verdier.
  Funksjonen kaller også på en annen funksjon, Indikator. Dette gjør den med
  input-en GreenLED.
******/
void bookingGodkjent(unsigned long booking_nr[], unsigned long booking_lengde){
  booking_nr[0] = millis();
  booking_nr[1] = booking_lengde*MIN_TIL_mS;
  booking_nr[2] = 1;
  Indikator(GreenLED);
}


/******
  Kalles når en booking er utgått. Tar inn to inputs og ingen outputs. Input-ene er
  booking_nr (eksempel KB1) og det tilhørende CoT signalet sin key. Funksjonen kaller
  på indikator funksjonen med pinnen tilegnet buzzer_LED. bookingTidUtaatt funksjonen
  setter CoT signalet til "0". Den reseter også alle leddene i booking nummerets
  array ([0]=[1]=[2]=0).
******/
void bookingTidUtaatt(unsigned long booking_nr[], char tilhoerende_signal_key[]){
  Indikator(buzzer_LED);
  circusESP32.write(tilhoerende_signal_key, 0, token);
  booking_nr[0] = 0;
  booking_nr[1] = 0;
  booking_nr[2] = 0;
}


/******
  Kalles for å indikerer at en booking er godkjent og registrert, avslått
  eller utgått. Funksjonen tar inn et input og gir ingen outputs. Input-en
  er pinnen funksjonen skal gjelde for. Funksjonen veksler mellom å sette
  denne pinne til høyverdi og laveverdi med 50ms mellomrom.
******/
void Indikator(int PIN){
  digitalWrite(PIN, HIGH);
  delay(50);
  digitalWrite(PIN, LOW);
  delay(50);
  digitalWrite(PIN, HIGH);
  delay(50);
  digitalWrite(PIN, LOW);
}
```