Bu projede, bebeğin uyku ve genel sağlık durumunu takip etmek, ebeveynlere canlı görüntüleme sağlamak ve beşiği otomatik kontrol etme özellikleri(otomatik sallama, ninni açma vb.) sunulmuştur. Proje, mobil bir uygulama (MIT App Inventor) ve entegre donanımlar (ESP8266, hareket sensörü, ısı sensörü ve DFPlayer Mini) ile gerçekleştirilmiştir.

Proje hedeflerimiz;

Bebek gözetimi ve sağlık takibini kolaylaştırmak.

Uyku kalitesi ve uyanma sıklığı analizleri sunmak.

Bebeğe ninni çalma ve gerektiğinde ebeveynlere bildirim gönderme.

Kullandığımız ekipmanlar

ESP8266 Modülü

ESP8266 Cam Modülü

Hareket sensörü

Isı sensörü (LM35)

DFPlayer-Mini ve hoparlör

Bağlantı kabloları ve dirençler

Servo Motor

Mobil Uygulama Yapısı

MIT App Inventor kullanılarak tasarlanan uygulama, aşağıdaki menü yapısına sahiptir:

Ana Sayfa

Kamera

Ninni İşlemleri

Analiz

Donanım:

ESP8266 modülü, hareket sensörü ve ısı sensörü ile entegre edilmiştir.

DFPlayer-Mini, mikro SD kartından ninni dosyaları oynatacak şekilde programlanmıştır.

Bağlantılar breadboard ve lehimleme yoluyla sağlanmıştır.

Yazılım:

Arduino Kodlama: Donanım kontrolü için gerekli olan kodlar Arduino IDE kullanılarak yazılmıştır. Kodda, ısı ve hareket algılandığında özgün senaryolar çalışmaktadır.

MIT App Inventor: Uygulama ekranı ve kontrol arayüzü, blok tabanlı kodlama kullanılarak geliştirilmiştir.

Arduino Kodlarımız:

Kamera Yayını İçin;

```
#include <WiFi.h>
#include "esp_camera.h"

const char* ssidAP = "Feyza";  // Wi-Fi ag adi
const char* passwordAP = "dain3415";       // Wi-Fi ag sifresi

void cameraInit(void);
void startCameraServer();


void setup() {
  Serial.begin(115200);  // Hata ayiklamak icin seri port ekran baslatildi
  Serial.setDebugOutput(true);
  Serial.println();

  cameraInit();  // Kamera konfigurasyonu yapildi

  WiFi.softAP(ssidAP, passwordAP);
  //  WiFi.softAPConfig(apip, gateway, subnet);
  delay(100);

  startCameraServer();  // Kamera server baslatildi

  Serial.print("Kamera hazir! Baglanmak icin 'http://");  // Baglanti saglandi
  Serial.print(WiFi.softAPIP());                          // Goruntunun
yayinlanacagi IP adresi seri port ekrane yaziliyor
  Serial.println("' adresini kullaniniz");
}

void loop() {
  delay(10000);
}

void cameraInit(void) {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = CAMD2;
  config.pin_d1 = CAMD3;
  config.pin_d2 = CAMD4;
  config.pin_d3 = CAMD5;
  config.pin_d4 = CAMD6;
  config.pin_d5 = CAMD7;
  config.pin_d6 = CAMD8;
  config.pin_d7 = CAMD9;
```

```
  config.pin_xclk = CAMXC;
  config.pin_pclk = CAMPC;
  config.pin_vsync = CAMV;
  config.pin_href = CAMH;
  config.pin_sscb_sda = CAMSD;
  config.pin_sscb_scl = CAMSC;
  config.pin_pwdn = -1;
  config.pin_reset = -1;
  config.xclk_freq_hz = 15000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG;
  //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  //init with high specs to pre-allocate larger
buffers                     for larger pre-allocated frame buffer.
  if (config.pixel_format == PIXFORMAT_JPEG) {
    if (psramFound()) {
      config.jpeg_quality = 10;
      config.fb_count = 2;
      config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
      // Limit the frame size when PSRAM is not available
      config.frame_size = FRAMESIZE_SVGA;
      config.fb_location = CAMERA_FB_IN_DRAM;
    }
  } else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
  }

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t* s = esp_camera_sensor_get();
  // Drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);
}
```

DFPlayer-Mini'den Ninni Çalması İçin;

```
#include <SoftwareSerial.h>
#include <DFRobotDFPlayerMini.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <HTTPClient.h>

// Wi-Fi Ayarları
const char* ssid = "Feyza";           // Wi-Fi SSID
const char* password = "dain3415";       // Wi-Fi şifresi

// MQTT Ayarları
const char* mqttServer = "broker.hivemq.com"; // MQTT broker adresi
const int mqttPort = 1883;                    // MQTT portu
const char* mqttTopicAlert = "baby_alert";    // Bebek ağlama bilgisi
const char* mqttTopicStatus = "lullaby_status"; // Ninni durumu

WiFiClientSecure client;
WiFiClient espClient;
PubSubClient mqttClient(espClient);
HTTPClient http; // HTTPClient nesnesi

// DFPlayer Mini Bağlantıları
SoftwareSerial mySerial(D8, D9); // RX = D2, TX = D3
DFRobotDFPlayerMini myDFPlayer;

// Ses Sensörü
const int soundSensorPin = D1;
bool isPlaying = false;

// Web sunucusundan alınacak dosya URL'si
const char* fileUrl =
"https://drive.google.com/uc?export=download&id=1_DqG0RoUzF9kerS2yj1ruCb-
jZfcEUQp"; // Burada kendi sunucunuzun linkini kullanın

void setup() {
  // Seri Haberleşme Başlatılıyor
  Serial.begin(115200);
  mySerial.begin(115200);

  // Ses Sensörü Pini
  pinMode(soundSensorPin, INPUT);
  pinMode(D9, INPUT);
  pinMode(D8, OUTPUT);

  // DFPlayer Mini Başlatma
  if (!myDFPlayer.begin(mySerial)) {
```

```
    Serial.println("DFPlayer Mini başlatılamadı!");
    while (true);
  }
  myDFPlayer.volume(30); // Ses seviyesi (0-30)

  // Wi-Fi Bağlantısı
  Serial.println("Wi-Fi'ye bağlanılıyor...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
  }
  Serial.println("Wi-Fi bağlantısı tamamlandı!");

  // MQTT Client Ayarları
  mqttClient.setServer(mqttServer, mqttPort);
  reconnectMQTT();
}

void loop() {
  // MQTT bağlantısı kesildiyse yeniden bağlan
  if (!mqttClient.connected()) {
    reconnectMQTT();
  }
  mqttClient.loop();

  // Ses Sensöründen Gelen Veriyi Oku
  int soundDetected = digitalRead(soundSensorPin);

  // Ses Algılandı ve Ninni Çalmıyorsa
  if (soundDetected == HIGH && !isPlaying) {
    Serial.println("Bebek ağlıyor!");
    mqttClient.publish(mqttTopicAlert, "Bebek ağlıyor!"); // MQTT mesajı
gönder
    playLullabyFromServer();  // Sunucudan ninniyi çal
    isPlaying = true;
    mqttClient.publish(mqttTopicStatus, "Ninni açık!"); // MQTT mesajı gönder
  }

  // Ninni Tamamlandıysa (myDFPlayer.available() kullanılarak kontrol edilir)
  if (isPlaying && !myDFPlayer.available()) {
    Serial.println("Ninni tamamlandı. Ses sensörü tekrar kontrol ediliyor.");
    isPlaying = false;
    mqttClient.publish(mqttTopicStatus, "Ninni kapalı!"); // MQTT mesajı
gönder
  }

  delay(100);
}
```

```
void reconnectMQTT() {
  while (!mqttClient.connected()) {
    Serial.println("MQTT'ye bağlanılıyor...");
    if (mqttClient.connect("ESP8266Client")) {
      Serial.println("MQTT bağlantısı başarılı!");
    } else {
      Serial.print("Bağlanılamadı, hata kodu: ");
      Serial.println(mqttClient.state());
      delay(2000);
    }
  }
}

void playLullabyFromServer() {
  Serial.println("Sunucudan ninni indiriliyor...");

  http.begin(client, fileUrl); // Sunucudan dosya URL'sine bağlan
  int httpCode = http.GET(); // Dosyayı indir

  if (httpCode == 200) {
    Serial.println("Dosya başarıyla indirildi!");
    // Burada, ses dosyasını geçici bir hafızaya indirip işleyebilirsiniz
    // Ancak DFPlayer Mini, sadece SD karttan okuma yapar, o yüzden MP3
dosyasını bir SD karttan okumanız gerekecek
  } else {
    Serial.println("Dosya indirilemedi!");
  }

  http.end();
}
```

Bebek Ağlamasını Algılayacak Ses Sensörü İçin;

```
#include <WiFi.h>
#include <HTTPClient.h>

// Wi-Fi ağ bilgileri
const char* ssid = "Feyza";          // Wi-Fi ağ adı
const char* password = "dain3415";       // Wi-Fi şifresi

// Ses sensörü pin tanımlaması
const int soundSensorPin = 30;  // Ses sensörü bağlantısı (örneğin, GPIO 34)

// HTTP sunucu bilgileri (MIT App Inventor)
const char* serverName = "http://192.168.136.206:8080/notify"; // MIT App
URL'si
```

```
void setup() {
  Serial.begin(115200); // Seri haberleşme başlatıldı
  delay(10);

  // Wi-Fi'ye bağlanma
  Serial.println("Wi-Fi'ye bağlanılıyor...");
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("Wi-Fi bağlantısı kuruldu!");
  Serial.print("ESP32'nin IP Adresi: ");
  Serial.println(WiFi.localIP());

  // Ses sensörünü giriş olarak ayarla
  pinMode(soundSensorPin, INPUT);
}

void loop() {

    // Bebek ağlıyor ise HTTP isteği gönder
    if(WiFi.status() == WL_CONNECTED) {
      HTTPClient http;

      http.begin(serverName);  // MIT App Inventor URL'sine bağlan
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");  //
Gönderilecek veri formatı

      String httpRequestData = "status=Bebek_agliyor";  // POST verisi
      int httpResponseCode = http.POST(httpRequestData);  // POST isteği
gönder

      // HTTP yanıtını kontrol et
      if (httpResponseCode > 0) {
        Serial.print("HTTP Response Code: ");
        Serial.println(httpResponseCode);
      } else {
        Serial.println("HTTP isteği başarısız oldu");
      }

      //http.end();  // HTTP bağlantısını sonlandır
    }

  delay(200); // 1 saniye bekle
}
```

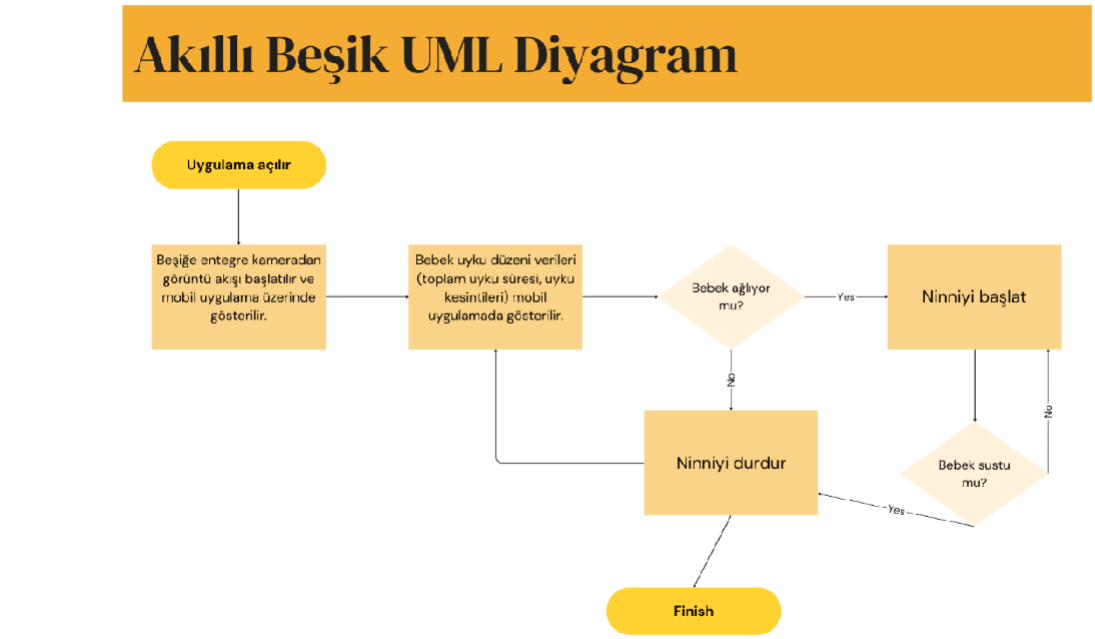
Maliyet Analizi:

Bileşen	*Adet*	*Fiyat*
ESP8266	*1*	*150*
Hareket Sensörü	*1*	*50*
Isı Sensörü	*1*	*30*
DFPlayer-Mini	*1*	*100*
Hoparlör	*1*	*70*
Bağlantı Kabloları	*10*	*30*
Servo Motor	*1*	*70*
Oyuncak Beşik	*1*	*150*
Toplam	*-*	*650*

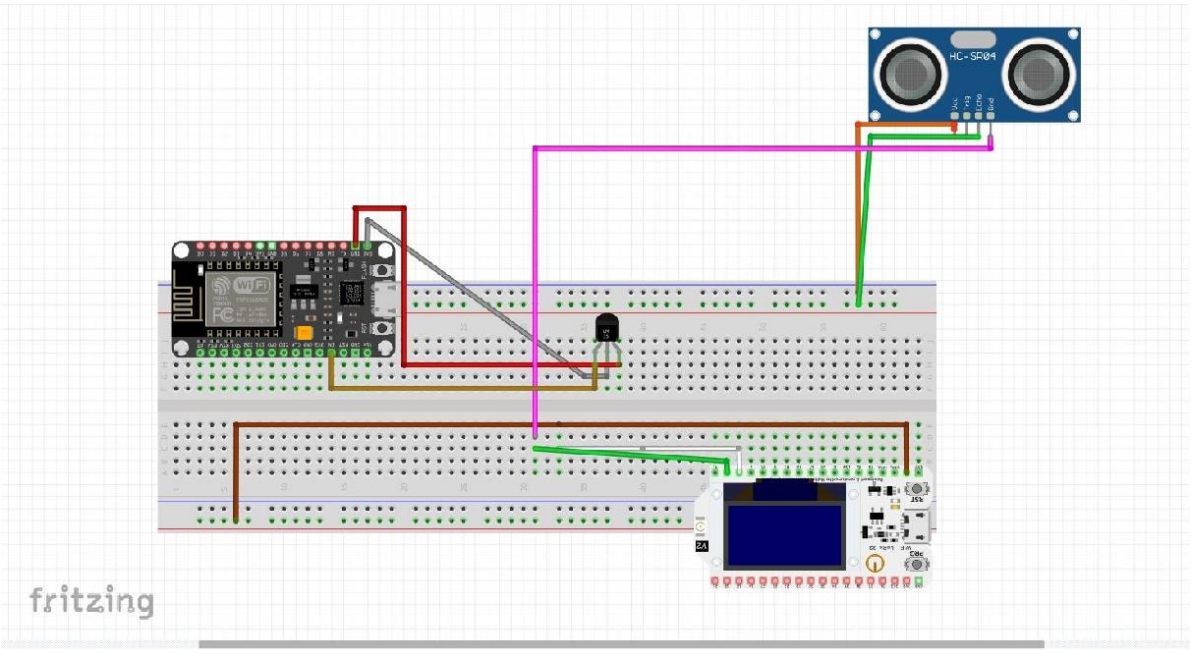
İş Modeli

Akıllı Beşik Modeli Kanvası

UML Diyagramı:



Fritzing Devre Tasarımı:



Sonuç:

"Uyku Arkadaşım" akıllı beşik uygulaması, ebeveynlerin bebeklerinin uyku ve sağlık durumlarını etkili bir şekilde takip etmelerini sağlamak için tasarlanmıştır. Uygulama, donanım ve yazılım uyumunu başarıyla sağlamış ve yenilikçi bir yaklaşım sunmuştur. İlerleyen süreçlerde daha fazla sensör eklenerek geliştirilebilir.