2015104124
전자공학과
진우빈

IoT 소프트웨어 과제12

예제 1)

키패드 모듈에서 받는 입력을 통해 모터를 제어하는 실습입니다. 키패드에 3개의 입력을 받아 각각 왼쪽으로 회전, 정지, 오른쪽 회전을 수행합니다. 아두이노의 9, 10번 핀에 모터모듈의 왼쪽회전과, 오른쪽회전을 할 수 있게 연결하였으며, 아두이노 메가 보드의 0,1,2 인터럽트 번호에 해당하는 2, 3, 21 핀번호에 키패드를 연결하였습니다.

xQueue라는 이름으로 Queue를 선언하고 Setup함수에서 3개의 인자 값을 넣을 수 있는 Queue를 만들어 주었습니다. 태스크 간 통신은 Queue를 사용하게 되는데 Queue는 xQueueCreate() 함수를 통해 만들어 집니다. 이 함수는 큐에 필요한 메모리 공간(uxQueueLength * uxItemSize)을 할당하고 그에 대한 queue handle(일종의 포인터)를 리턴 해 줍니다.

xQueueSendFromISR함수를 통해 입력 값을 다른 태스크로 보낼 수 있고 xQueueReceive를 통해 다른 태스크로부터 값을 받을 수 있습니다. 두 함수 모두 첫 번째 파라미터는 queue handle(xQueueCreate에서 리턴된 값)이고, 두 번째 파라미터는 집어넣을 아이템(데이터)에 대한 포인터 또는 읽어 올 아이템을 넣을 버퍼의 포인터입니다. 두 함수 모두 데이터를 복사하는 동작을 수행하게 됩니다. 세 번째 파라미터는 동작이 완료될 때 까지 기다리는 시간입니다. '0'으로 설정하면 함수는 아이템을 큐에 쓰거나 큐에서 읽어 오는 동작이 완료되었는가 여부에 관계없이 바로 리턴하게 됩니다.

LeftKeyControl, StopKeyControl, RightKeyControl은 xQueueSendFromISR 함수를 통해 각각의 경우에 따라 SendValue값을 xQueue로 보냅니다. 1번은 왼쪽회전 2번은 정지, 3번은 오른쪽회전을 할 수 있게끔 SendValue값을 보내게 됩니다. 그리고 MotorTask 함수에서는 xQueueReceive함수를 통해 받은 값에 따라 왼쪽, 오른쪽 모터의 출력을 제어하게 됩니다. 이후 Setup함수에서 모터에 연결된 핀은 출력, 키패드에 연결된 핀은 입력으로 설정하였고 인터럽트 번호에 맞게끔 attachInterrupt를 통해 인터럽트를 설정하였습니다.

멀티태스킹 실습에서 했던 것처럼 Motor를 제어하는 MotorTask함수 수행하는 태스크를 xTaskCreate로 만들어주었고 성공적으로 태스크를 만들었습니다.

컴파일 후 실행결과 1번을 누르면 왼쪽으로 모터가 돌아가게되고 2번을 누르면 정지, 3번을 누르면 오른쪽으로 모터가 돌아가는 것을 확인할 수 있었습니다.

```
void LeftKeyControl() {
  uint16_t sendValue = 1;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void StopKeyControl() {
  uint16_t sendValue = 2;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void RightKeyControl() {
  uint16_t sendValue = 3;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void MotorTask(void* arg) {
  uint16_t receiveValue = 0;

  while(1) {
    if(xQueueReceive(xQueue, &receiveValue,0)) {
      //left
      if(receiveValue == 1) {
        digitalWrite(MT_P, LOW);
        digitalWrite(MT_N, HIGH);
      }

      //Stop
      else if(receiveValue == 2) {
        digitalWrite(MT_P, LOW);
        digitalWrite(MT_N, LOW);
      }

      //right
      else if(receiveValue == 3) {
        digitalWrite(MT_P, HIGH);
        digitalWrite(MT_N, LOW);
      }
    }
  }
```

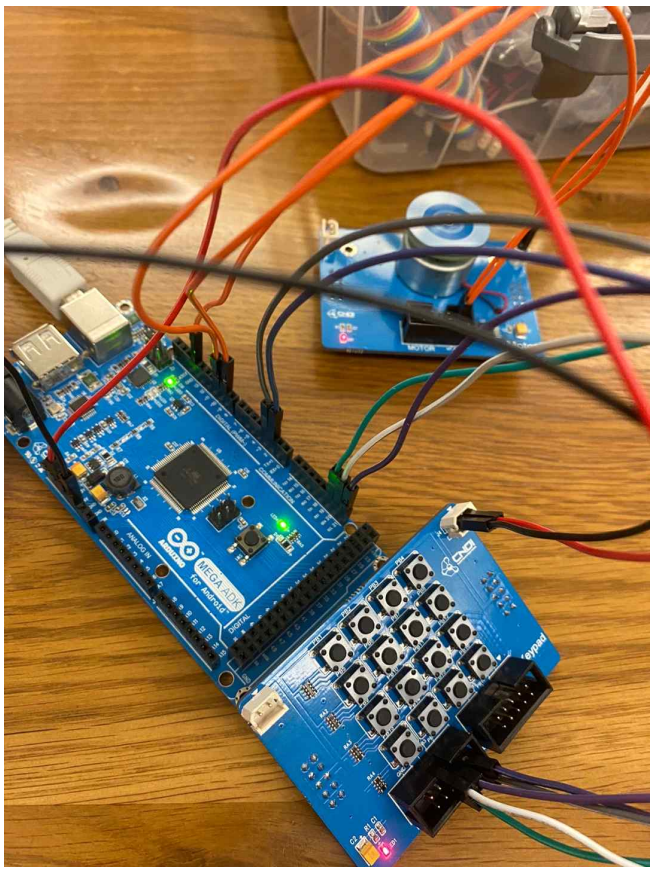
```
void setup() {
  pinMode(MT_P, OUTPUT);
  pinMode(MT_N, OUTPUT);
  pinMode(LeftKey, INPUT);
  pinMode(StopKey, INPUT);
  pinMode(RightKey, INPUT);

  attachInterrupt(0, LeftKeyControl, RISING);
  attachInterrupt(1, StopKeyControl, RISING);
  attachInterrupt(2, RightKeyControl, RISING);

  xQueue = xQueueCreate(3, sizeof(uint16_t));

  if(xQueue != NULL) {
    xTaskCreate(MotorTask, NULL, 200, NULL, 1, NULL);
    vTaskStartScheduler();
  }
}
```

<lab3-3_1.c 코드>

과제 1)

예제 1에서 추가로 입력을 2개 더 받아서 왼쪽 모터회전을 가속하거나 오른쪽 모터회전을 가속하는 기능을 추가하는 실습입니다. 예제 1과 전체적인 틀은 비슷하나 Keypad에서 두 개의 입력을 더 받아야하기 때문에 인터럽트 핀 번호인 0,1,2에 더해서 3, 4에 대응되는 20번과 19번을 키패드 모듈에 연결하였습니다. 이후 LeftKeyAccerelate, RgithKeyAccerelate 함수를 만들어서 sendValue값 4, 5를 Motor를 돌리는 태스크의 함수로 전달할 수 있게 설정하였습니다. MotorTask함수에는 receive된 인자가 4, 5일 때 각각 왼쪽으로 가속, 오른쪽을 가속할 수 있게하였습니다. 이 때 pwm을 사용하여야 하는데 analogWrite함수를 사용하여 pwm을 제어하였습니다. DutyCycle이 100%일 때는 255이므로 각각 50% 75%, 100%에 해당하는 값을 크기 3의 DutyCycle배열에 넣어주었습니다. 그리고 모터를 처음 회전시킬 때의 DutyCycle인 25%로 모터가 돌아갈 때는 64의 값으로 설정을 하였습니다. 이후 Setup함수에서 각각 입출력을 설정하고 큐에 필요한 메모리공간만큼의 크기를만들어 xQueueCreate를 통해 xQueue를 만들었습니다.

태스크를 만들고 실행 결과 모터가 원활히 돌아가지 않았습니다. 가속은 잘 되었으나 25% Cycle에서 모터 출력전압이 부족한 탓인지는 모르겠으나 원활히 돌아가지 않고 직접 손으로 힘을 주어야만 힘들게 돌아가는 것을 확인하였습니다. 따라서 애초의 25%로 설정한 DutyCycle값을 64가아니라 80정도로 수정하여 혼자 힘으로도 모터가 잘 돌아갈 수 있게 하였습니다. 이후 키패드를 눌렀을 때 1, 3번일 때 80/255 Duty Cycle로 각각 왼쪽 오른쪽으로 회전하고 2번일 때 정지 그리고 각각 가속이 잘 되는 것을 확인하였습니다.



<아두이노에 키패드와 모터 모듈을 연결한 사진>

```
// 아두이노 인터럽트 핀번호0~4에 해당하는 번호로 설정
const int LeftKey = 2;
const int StopKey = 3;
const int RightKey = 21;
const int LeftAcc = 20;
const int RightAcc = 19;

// 25% Duty Cycle 정의 64로 할 시 전원이 부족해서 인지 모터가 제대로 돌아가지 않아서 70으로 설정
// 50%, 75%, 100% 순서대로의 Duty Cycle 값
const int Duty25 = 85;
const int DutyCycle[3]={127,191,255};
int CycleNumLeft = -1;
int CycleNumRight = -1;
```

```
void LeftKeyControl() {
  uint16_t sendValue = 1;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void StopKeyControl() {
  uint16_t sendValue = 2;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void RightKeyControl() {
  uint16_t sendValue = 3;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void LeftKeyAccelerate() {
  uint16_t sendValue = 4;
  xQueueSendFromISR(xQueue, &sendValue,0);
}

void RightKeyAccelerate() {
  uint16_t sendValue = 5;
  xQueueSendFromISR(xQueue, &sendValue,0);
}
```

<xQueueSendFromISR을 통해 태스크로 인자를 전달하는 함수 및 설정한 Duty Cycle>

```
void MotorTask(void* arg) {
  uint16_t receiveValue = 0;

  while(1) {
    if(xQueueReceive(xQueue, &receiveValue,0)) {
      //left rotate
      if(receiveValue == 1) {
        CycleNumLeft = -1;
        CycleNumRight = -1;
        digitalWrite(MT_P, LOW);
        analogWrite(MT_N, Duty25);
      }

      //Stop
      else if(receiveValue == 2) {
        CycleNumLeft = -1;
        CycleNumRight = -1;
        digitalWrite(MT_P, LOW);
        digitalWrite(MT_N, LOW);
      }

      //right rotate
      else if(receiveValue == 3) {
        CycleNumLeft = -1;
        CycleNumRight = -1;
        analogWrite(MT_P, Duty25);
        digitalWrite(MT_N, LOW);
      }

      //left Accelerate
      else if(receiveValue == 4) {
        CycleNumRight = -1;
        CycleNumLeft=(CycleNumLeft+1)%3;
        digitalWrite(MT_P, LOW);
        analogWrite(MT_N, DutyCycle[CycleNumLeft]);
      }

      //right Accelerate
      else if(receiveValue == 5) {
        CycleNumLeft = -1;
        CycleNumRight=(CycleNumRight+1)%3;
        analogWrite(MT_P, DutyCycle[CycleNumRight]);
        digitalWrite(MT_N, LOW);
      }
    }
  }
```

<왼쪽으로 가속, 오른쪽으로 가속하는 기능을 추가한 MotorTask 함수>