Init

캡쳐 세션 생성

- session = [[AVCaptureSession alloc] init];
- session.sessionPreset = AVCaptureSessionPresetPhoto;

캡쳐 기기 미디어 타입 선정

- AVCaptureDevice *device = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

캡쳐 세션 인풋(앞에 선언한 기기 정보) 적용

- AVCaptureDeviceInput *input = [AVCaptureDeviceInput deviceInputWithDevice:device error:**nil**];
- [session addInput:input];

결과값 바인딩 변수 선언 및 세션에 할당

- output = [[AVCaptureVideoDataOutput alloc];
- [session addOutput:output];
- output.videoSettings = @{ (NSString *)kCVPixelBufferPixelFormatTypeKey : @(kCVPixelFormatType_32BGRA) };

처리방식

Timer를 이용해 0.2 초마다 캡쳐 결과값을 확인하다.

캡쳐한 값을 세션의 아웃풋에서 꺼내오는 작업을 진행할 수 있도록 Queue를 선언하여 output에 할당하는 snapshot 함수 0.2초 마다 실행

- timer = [NSTimer scheduledTimerWithTimeInterval:0.2 target:**self** selector:**@selector**(snapshot) userInfo:**nil** repeats:**YES**];

세션 시작

- [session startRunning];

Snapshot 함수

```
- (UIImage *)snapshot {
   //캡쳐 시작
   if(dispatchFlag == false){
       sampleBufferCallbackQueue = dispatch_queue_create("VideoDataOutputQueue", DISPATCH_QUEUE_SERIAL);
       [output setSampleBufferDelegate:self queue:sampleBufferCallbackQueue];

       dispatchFlag = true;
   }

   count = count +1;
```

```
    UIImage *image = nil;
    return image;
}
```

———————————— 해당 부분 매우 불필요해 보임————————————

캡쳐 데이터가 버퍼에 들어왔을때 자동으로 작동하는 델리게이트 작성

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput
didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer fromConnection:
(AVCaptureConnection *)connection{
    capturedImage = [self imageFromSampleBuffer:sampleBuffer];
}
```

버퍼에 있는 데이터를 Image화 시키는 작업

```
- (UIImage *)imageFromSampleBuffer:(CMSampleBufferRef) sampleBuffer {
    CVImageBufferRef imageBuffer =
CMSampleBufferGetImageBuffer(sampleBuffer);
    CVPixelBufferLockBaseAddress(imageBuffer, 0);

    void *baseAddress = CVPixelBufferGetBaseAddress(imageBuffer);
    size_t bytesPerRow = CVPixelBufferGetBytesPerRow(imageBuffer);
    size_t width = CVPixelBufferGetWidth(imageBuffer);
    size_t height = CVPixelBufferGetHeight(imageBuffer);

    CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
    CGContextRef context = CGBitmapContextCreate(baseAddress, width,
height, 8, bytesPerRow, colorSpace, kCGBitmapByteOrder32Little |
kCGImageAlphaPremultipliedFirst);

    CGImageRef quartzImage = CGBitmapContextCreateImage(context);
    CVPixelBufferUnlockBaseAddress(imageBuffer,0);

    CGContextRelease(context);
    CGColorSpaceRelease(colorSpace);

    UIImage *image =  [UIImage imageWithCGImage:quartzImage scale:1.0
orientation:UIImageOrientationRight];

    CGImageRelease(quartzImage);

    return (image);
}
```

이미지 결과 바인딩

```
#pragma mark - CLICK - 카메라 촬영
- (IBAction)onClickCamera:(id)sender {
// 기존에 실행되던 타이머와 캡처에 이용되는 세션 종료
    [timer invalidate];
    [session stopRunning];
    NSLog(@"## CAPTURE : %@", capturedImage);
// 이미지 크롭핑 진행
    CGFloat topcrop = _vMiddleShadow.frame.origin.y/
SCREEN_HEIGHT*capturedImage.size.height;

    CGRect cropRect = CGRectMake(topcrop, capturedImage.size.width/5,
capturedImage.size.height-(2*topcrop), capturedImage.size.width/5*3);

    CGImageRef imageRef = CGImageCreateWithImageInRect([capturedImage
CGImage], cropRect);

    croppedImage = [UIImage imageWithCGImage:imageRef scale:1.0
orientation:UIImageOrientationRight];
    CGImageRelease(imageRef);

    }
```